НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ С. Г. СТІРЕНКО

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Система віртуальної реальності для визначення дій людини у лабіринті»

Виконав (-ла):
студент IV курсу, групи ІП-64
Каленик Павло Іванович _____

Керівник:
доцент, к.т.н.
Волокита Артем Миколайович _____

Консультант з нормо контролю:
Професор, доктор технічних наук
Сімоненко Валерій Павлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
"___"_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Каленику Павлу Івановичу

1. Тема проєкту «*Система віртуальної реальності для визначення дій людини у лабіринті»*

керівник проєкту Волокита Артем Миколайович, доцент, к.т.н., затверджені наказом по університету від *«07»* травня __2020р. №__1081-с__

2. Термін здачі студентом закінченого роботи__*27 травня 2020р.*
3. Вихідні дані до проекту*: технічна документація, теоретичні та статистичні дані.*
4. Зміст пояснювальної записки:

   Розділ 1. Огляд віртуальної реальності та аналіз існуючих систем

   Розділ 2. Огляд технологій для розробки системи

   Розділ 3. Деталі розробки системи

Розділ 4. Аналіз розробленої системи

5. Консультант роботи, з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1	доцент, к.т.н. Волокита А. М.		
Розділ 2	доцент, к.т.н. Волокита А. М.		
Розділ 3	доцент, к.т.н. Волокита А. М.		
Розділ 4	доцент, к.т.н. Волокита А. М.		

6. Дата видачі завдання *14.12 .2019 року*

КАЛЕНДАРНИЙ ПЛАН

п/п	Найменування етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту(роботи)	Примітки
1.	Затвердження теми роботи	14.12.2019-19.12.2019	Виконано
2.	Вивчення та аналіз завдання	19.12.2019-19.03.2020	Виконано
3.	Розробка архітектури та загальної структури системи	19.03.2020-30.03.2020	Виконано
4.	Розробка структур окремих підсистем	30.03.2020-10.04.2020	Виконано
5.	Програмна реалізація системи	10.04.2020-20.04.2020	Виконано
6.	Оформлення пояснювальної записки	15.04.2020-10.05.2020	Виконано
7.	Захист програмного продукту	25.04.2020	Виконано
8.	Перед захист	28.05.2020	Виконано
9.	Захист	19.06.2020	Виконано

Студент Павло КАЛЕНИК _____

(підпис)

Керівник Артем ВОЛОКИТА_____

(підпис)

АНОТАЦІЯ

У роботі розроблено систему віртуальної реальності для визначення дій людини у лабіринті. Дана система дозволяє занурити людину у віртуальний лабіринт та перевірити, як людина орієнтується у віртуальному просторі. В даній системі було реалізовано два варіанти переміщення користувача по віртуальному лабіринту.

Під час проходження лабіринту, система вираховує де саме знаходився користувач і скільки разів. Ці дані система зберігає на пристрої користувача. Система зберігає дані часу, за який користувач зміг знайти вихід із лабіринту.

Також, в системі реалізована відеозапис проходження користувача, яка записує усі дії користувача під час проходження лабіринту. Всі ці дані користувач може звірити та проаналізувати. З отриманих даних можна зробити висновок чи схильна людина до хвороби Альцгеймера. У системі також реалізована гра для тренування пам’яті користувача.

Ключові слова: віртуальна реальність, віртуальний лабіринт, гра на память, android, хвороба Альцгеймера.

ABSTRACT

There was developed the virtual reality system to determine human actions in a maze. This system allows to immerse a person in a virtual maze and check how a person navigates in cyberspace. In this system, two options for moving the user through a virtual maze were implemented.

During the passage of the maze, the system calculates exactly where the user was and how many times. The system stores this data on the user's device. The system stores data of the time during which the user was able to find a way out of the maze.

Also, the system implements a video of the user's passage, which records all the user's actions during the passage of the maze. All this data can be compared and analyzed by the user. From the data obtained, it is possible to conclude whether a person is prone to Alzheimer's disease. The system also implements a game to train user memory.

Key words: virtual reality, virtual maze, memory game, android, Alzheimer's disease.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
	А4		Завдання на дипломний проєкт		
	А4	ІАЛЦ.467200.001 ВП	Відомість проєкту	1	
	А4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	3	
	А4	ІАЛЦ.467200.003 ПЗ	Пояснювальна записка	96	
	А3	ІАЛЦ.467200.004 Д1	Алгоритм дій програмного забезпечення	1	
	А3	ІАЛЦ.467200.005 Д2	Функціональна схема (діаграма класів)	1	
	А3	ІАЛЦ.467200.006 Д3	Структурна схема системи	1	
	А4	ІАЛЦ.467200.007 Д4	Текст програмного коду	13	

					ІАЛЦ.467200.001 ВП		
Зм.	Арк.	№ документна	Підпис	Дата			
Розробив		Каленик П.І.			Система віртуальної реальності для визначення дій людини у лабіринті	Літ.	Аркуш 1 / Акрушів 1
Перевірив		Волокита А.М.					
Реценз.					Відомість дипломного проекту	НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІП-64	
Н. Контр.		Сімоненко В.П.					
Затверд.							

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЕКТУ

на тему: «*Система віртуальної реальності для визначення дій людини у лабіринті*»

Київ – 2020

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ 2

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ 2

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ 2

4 ДЖЕРЕЛА РОЗРОБКИ 2

5 ТЕХНІЧНІ ВИМОГИ 2

5.1. Вимоги до програмного продукту, що розробляється 2

5.2. Вимоги до інструментального програмного забезпечення 3

5.3. Вимоги до апаратної частини обчислювальної системи 3

6 ЕТАПИ РОЗРОБКИ 3

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Технічне завдання стосується розробки системи віртуальної реальності для визначення дій людини у лабіринті.

Областю застосування даної системи є перевірка орієнтації людини у віртуальному просторі та збереження її результату проходження.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою і призначенням даної роботи є створення системи віртуальної реальності, яка буду відстежувати пересування людини у лабіринті для перевірки орієнтації у просторі.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки дипломного проекту є науково-технічна література, офіційні документації технологій, які використовувались для розробки системи, публікації в мережі Інтернет по даній темі.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

Розроблена система має виконувати наступні вимоги:

1) Надати зручний користувацький інтерфейс для кращого досвіду взаємодії користувача з віртуальною реальністю.
2) Збір та зберігання даних переміщення користувача по віртуальному лабіринту.
3) Можливість вибору варіанту переміщення у головному меню.
4) Можливість робити відеозапис проходження лабіринту.
5) Надавати можливість для тренування пам’яті користувачу.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux;
- Unity**;**
- C# 7.3;
- Microsoft Visual Studio 2019.

5.3. Вимоги до апаратної частини

- ОС Android 5.0 або новіша;
- Окуляри віртуальної реальності Cardboard або інші сумісні зі смартфоном;
- Вільної пам’яті на смартфоні більше ніж 63 МБ;

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	
Вивчення та аналіз завдання	
Розробка архітектури та загальної структури системи	
Розробка структур окремих частин системи	
Програмна реалізація системи	
Виправлення помилок	
Оформлення пояснювальної записки	
Передзахист	
Захист	

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Система віртуальної реальності для визначення дій людини у лабіринті*»

Київ – 2020

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 3

ВСТУП 4

РОЗДІЛ 1 ОГЛЯД ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ ТА ІСНУЮЧИХСИСТЕМ.. 7

1.1 Види реальності 7

1.2 Історія віртуальної реальності 9

1.3 Тенденції розвитку віртуальної реальності 10

1.4 ВР на світовому ринку 13

1.5 Відмінність додатків для ПК і для окулярів ВР 14

1.6 Що таке імерсивність та для чого необхідне? 16

1.7 Пристрій окуляри VR та різниця між окулярами VR 17

1.8 Взаємодія і системи трекінгу пристроїв 21

1.9 Існуючі системи віртуальної реальності та їх аналіз 24

1.10 Розгляд хвороби Альцгеймера 27

ВИСНОВОК ДО РОЗДІЛУ 1 29

РОЗДІЛ 2 ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ 30

2.1 Стаціонарний та мобільний VR 30

2.2 Пристрої управління VR 32

2.3 Unity. Що це та для чого? 34

2.4 VR та Unity 42

2.5 Google VR SDK 44

2.6 C#, .NET, Mono 49

2.7 Підтримка профілю .NET в Unity 53

ВИСНОВОК ДО РОЗДІЛУ 2 55

РОЗДІЛ 3 ДЕТАЛІ РОЗРОБКИ СИСТЕМИ 56

3.1 UI у системі ВР 56

3.2 Сцени та головні об’єкти системи. 58
3.3 Варіанти переміщення користувача та їх розробка 59
3.4 Розробка точок для переміщення 61
3.5 Розробка UI для пересування 65
3.6 Розробка UI для повернення на початок лабіринту 67
3.7 Розробка UI для відеозапису 68
3.8 Розробка гри для тренування пам’яті користувача 70
ВИСНОВОК ДО РОЗДІЛУ 3 72
РОЗДІЛ 4 АНАЛІЗ РОЗРОБЛЕННОЇ СИТЕМИ 73
4.1 Як взаємодіяти із системою 73
4.2 Взаємодія з головним меню 73
4.3 Взаємодія зі сценою з першим варіантом переміщення 76
4.4 Взаємодія з другим варіантом переміщення 81
4.5 Взаємодія з грою для тренування пам’яті 82
ВИСНОВОК ДО РОЗДІЛУ 4 85
ВИСНОВКИ 86
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 87
ДОДАТОК 1
ДОДАТОК 2
ДОДАТОК 3
ДОДАТОК 4

ПЕРЕЛІК СКОРОЧЕНЬ

VR – virtual reality.

ВР – віртуальна реальність.

AR – augmented reality, доповнена реальність.

MR – mixed reality, змішана реальність.

Tracking – система трекінгу для відстеження місця розташування і руху користувача у віртуальному просторі.

SDK – Software Development Kit, набір засобів розробки, який дозволяє розробникам програмного забезпечення створювати додатки для визначеного пакету програм, програмного забезпечення базових засобів розробки, апаратної платформи, комп'ютерної системи, ігрових консолей, операційних систем і інших платформ.

UI – User Interface, користувацькій інтерфейс, інтерфейс, який забезпечує передачу інформації між користувачем і програмно-апаратними компонентами комп'ютерної системи, використовується для взаємодії користувача із системою.

GVR – Google Virtual Reality, позначення для об'єктів, які містяться у пакеті Google VR SDK, такі як префаби, скрипти та інше.

App – application, програмне забезпечення, це програма або група програм, розроблена для кінцевих користувачі.

APK - Android Package, це формат файлу, який використовується операційною системою Android для встановлення мобільних додатків, мобільних ігор та програмного забезпечення.

API - Application Programming Interface, прикладний програмний інтерфейс, набір компонентів, якими одна програма взаємодіє з іншою.

ОС – операційна система.

OS – operation system.

3D – Three-dimensional space, тривимірний простір.

ВСТУП

Віртуальна реальність (з англ. Virtual Reality) зараз займає важливе місце у світі. Люди використовують її у різних сферах життя, таких як освіта, маркетинг, архітектура і дизайн, промисловість, медицина та у багатьох інших галузях. Зараз віртуальну реальність використовують не тільки для забави, а для допомоги людям у різних сферах життя.

Технологія віртуальної реальності зараз посідає велике місце в медицині. Системи віртуальної реальності, які зараз існують, допомагають, як хворим людям, так і лікарям. Віртуальна реальність це дуже потужна технологія, оскільки вона дає можливість занурити людину у віртуальний простір, дати їй відчуття реальної присутності, що поки не дають такої можливості інші технології.

Віртуальну реальність використовують для виявлення у людей схильності до хвороби Альцгеймера. Хвороба Альцгеймера - є найбільш частою причиною деменцій у похилому і старечому віці – її діагностують у 50% хворих з деменцією [1]. Порушення пам’яті є першою і більш значною скаргою [1].

Існують системи віртуальної реальності, які перевіряють як людина орієнтується у навколишньому просторі. Проте, ці системи не є доступними для великої кількості людей. Ця хвороба уражає, як людей старого віку, так і молодшого віку. Потрібно продовжувати створювати системи віртуальної реальності для визначення хвороби Альцгеймера, особливо, для мобільних пристроїв, оскільки в такому випадку така система буде доступною для максимально великої кількості користувачів.

Тому є необхідність у створенні системи віртуальної реальності для визначення дій людини у лабіринті. Така система має занурювати користувача у віртуальний простір та визначати, як людина орієнтується у цьому просторі. Така система повинна бути максимально доступною для великої кількості

людей, а тому має бути створена для мобільного ВР. Також, у такій системі повинна бути можливість для тренування пам’яті користувача.

Об’єктом дипломної роботи є система віртуальної реальності для визначення дій людини у лабіринті. Вибір об’єкту для створення зумовлений відсутністю системи віртуальної реальності для мобільних пристроїв з операційною системою Android, для визначення переміщення людини у віртуальному лабіринті, для відеозапису поведінки користувача, а також для можливості тренування своєї пам’яті користувачеві.

Метою роботи є:

1) Повний огляд ВР, види реальності, відмінності комп’ютерних додатків від додатків ВР, огляд існуючих систем ВР для допомоги людям у різних сферах життя та їх порівняння з моєю системою, огляд пристроїв ВР, ергономіка, імерсивність, системи трекінгу.

2) Огляд мобільної ВР та розгляд хвороби Альцгеймера.

3) Огляд технології створення програмного забезпечення віртуальної реальності, що необхідно враховувати при створені системи ВР.

4) Створення системи віртуальної реальності для визначення дій людини у лабіринті для мобільних пристроїв.

Були поставлені наступні завдання:

1. Детальний огляд ВР та аналіз існуючих систем.
2. Огляд технологій для створення систем ВР.
3. Створення системи віртуальної реальності.
4. Аналіз створеної системи.

Дипломна робота складається зі вступу, трьох розділів, висновків до кожного із розділів та загальних висновків. У першому розділі детально оглянуто: види віртуальної реальності, існуючі системи віртуальної реальності, історія віртуальної реальності, платформи ВР, імерсивність, системи трекінгу та інше.

У другому розділі розглянуті необхідні технології для створення системи віртуальної реальності, такі як ігровий рушій Unity, пакет Google VR SDK, мова програмування C#, платформа Mono та інші.

У третьому розділі описуються деталі розробки створеної системи**.** У четвертому розділі проводиться аналіз створеної системи, її взаємодія між користувачем.

РОЗДІЛ 1
ОГЛЯД ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ ТА ІСНУЮЧИХСИСТЕМ

1.1 Види реальності

Існує віртуальна, доповнена та змішана реальність.

Віртуальна реальність

Віртуальна реальність (з англ. Virtual Reality) - це створений технічними засобами світ, який передається людині через її відчуття [2]. Є різні платформи ВР, які існують на ринку десятки років, важливим з яких є шолом віртуальної реальності. Під час роботи на звичайному комп'ютері, людина сприймає інформацію через рамку екрану. Тобто всі дані, все зображення обмежено нею. Але, перебуваючи в окулярах ВР, людина не дивиться на зображення, вона фактично знаходиться всередині нього. Тобто, створюється так званий ефект присутності всередині віртуального середовища. У ВР людина сприймає штучне середовище.

Рисунок 1.1. Взаємодія користувача з окулярами ВР [3].

Доповнена реальність

Доповнена реальність (від англ. augmented reality) – це реальність в якій віртуальні об'єкти доповнюють об’єктивну дійсність. Тобто, вони інтегруються у те, що людина бачить навколо себе [4].

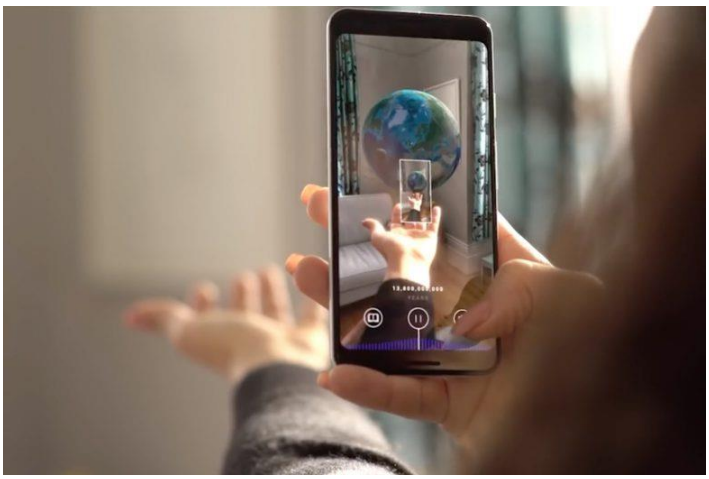


Рисунок 1.2. Взаємодія користувача з доповненою реальністю [5].

Змішана реальність

Змішана реальність (від англійського mixed reality) - це проектування 3D віртуальних об'єктів або голограм на фізичний простір, що дозволяє переміщатися навколо віртуального 3D об'єкта, оглядати його з усіх боків та, навіть, всередині [4].

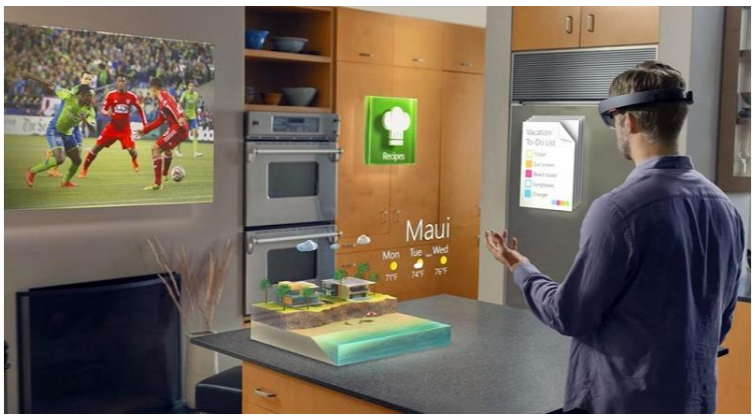


Рисунок 1.3. Взаємодія користувача з змішаною реальністю [6].

Пристрої через які людина сприймає навколо себе об'єкти AR та MR пройшли досить тривалий шлях, починаючи з Google Glass і закінчуючи сучасним пристроєм HoloLens (від компанії Microsoft). Також на ринку з'являється багато пристроїв для AR та VR.

Існує десктопний (Oculus Rift, HTC Vive, Playstation VR, Valve Index) та мобільний варіант віртуальної реальності (Google Cardboard, Samsung Gear, Google Daydream), а також існує автономний варіант BP Oculus Go.

1.2 Історія віртуальної реальності

Віртуальна реальність з'явилася в кінці 50-х років, почали з'являтися перші розробки, які були досить масивні і складно було зрозуміти, як вони можуть бути застосовні на той момент [7].

Після 60-70-х роках почали з'являтися вже комерційні пристрої, які люди починали використовувати в бізнесі, наприклад, для тренування пілотів, потім вже почали з'являтися такі комерційні системи, які дозволяли моделювати певні ситуації для тренувань цих людей.

Рисунок 1.4. Тренування пілотів за допомогою ВР [8].

Після цього, коли у 90-х роках віртуальна реальність з'явилася на ігровому ринку, з'явилося досить багато різних проектів віртуальної реальності, але, на жаль, всі вони теж не були успішні, тому що була висока вартість пристроїв, низька якість зображень та було дуже мало контенту.

Сильний виток технологій відбувся у 2012 році, з'явився шолом Oculus Rift DK1 [9], який досить швидко вплинув на те, що почали з'являтися інші рішення віртуальної реальності на ринку. Зараз найбільші корпорації зайняті в цьому бізнесі, йде божевільна гонка, віртуальна реальність проникла в мобільні рішення, створюються нові персональні комп'ютери, створюються нові шоломи віртуальної реальності. Щодня з'являються нові команди розробників, вони шукають де віртуальну реальність можна застосувати, які у віртуальній реальності є патерни, в яких сферах людини вона може бути корисною [10].

1.3 Тенденції розвитку віртуальної реальності

Існує безліч прогнозів, куди рухається віртуальна реальність, куди розвивається дана технологія, і всі ці прогнози пов'язані з двома великими цілями [11].

Перша ціль - це прихід ВР в mainstream, тобто коли ВР набуває масової популярності, а також використання ВР, наприклад, у бізнесі [12].

Друга ціль - це поліпшення користувачам користувацького досвіду, створення більш глибокого, потужного ефекту занурення. Всі ці цілі перетинаються між собою [12]

Для досягнення цих цілей в близькому майбутньому необхідно вирішити декілька моментів.

Перший момент - це скорочення вартості обладнання (віртуальних окулярів та іншої гарнітури). Вартість на даний момент є однією з головних проблем на шляху до популярності ВР. Також, при цьому дуже важливо покращувати якість основних характеристик пристроїв даних продуктів [13].

Якщо порівнювати з шоломом віртуальної реальності з 90-х років, то звичайно ж, вони пішли далеко вперед, але сучасним рішенням є ще куди прагнути. Легкість, дозвіл екрану, щільність пікселів, а також ширина кута огляду - це кращий ефект занурення користувачів у віртуальний простір.

Також, окремий випадок поліпшення характеристик є ергономіка, тобто зручність і практичність пристрою. Тобто, пристрій повинен зручно сидіти на голові, він повинен дуже легко налаштовуватися, він повинен бути легким, за рахунок чого в ньому можна було б перебувати довше часу.

Також, є одна досить істотна проблема з дротами, дуже багато користувачів скаржаться на те, що дроти заважають їм під час руху в шоломах, вони постійно заплутуються в них, коли намагаються повернутися, а всередині ВР це негативно впливає на призначений для користувача досвід. Зараз ведуться роботи над бездротовими рішеннями ВР, таких платформ як HTC Vive. Ще один вектор розвитку - це система управління трекінгу у ВР [14].

Tracking - відстеження місця розташування і руху користувача. Є різні способи взаємодії з віртуальним простором, а саме люди можуть управляти об'єктами та оточенням у них. Деякі системи управління здаються найбільш ефективними, але їх також можна зробити набагато краще і зараз багато уваги приділяється моушен контролерам (motion controllers) [15].

Motion controllers - це пристрої, які користувач тримає в руці і за допомогою яких користувачі управляють об’єктами у віртуальному просторі. Дані пристрої продовжать вдосконалюватися в майбутньому, як на рівні трекінгу так і в плані ергономіки [16].

Рисунок 1.5. Приклад Motion controllers [17].

Eye tracking – це технологія, в якій за допомогою спеціальних камер відстежується рух зіниць користувача. Це дозволяє спростити багато в чому управління, користувач може легко сфокусуватися на об'єкті віртуального простору. Крім цього технологія трекінгу зможе використовуватися в рамках так званого методу Foveated rendering [18] .

Рисунок 1.6. Відстеження зіниці технологією Eye tracking [19].

Foveated rendering – процес, коли частина зображення в центрі поля зору користувача відображується у повній якості, а решта розмивається в міру

наближення до периферії. Тобто, коли відбувається рендерінг тільки в тій частині, куди дивляться очі користувача. Це дозволить скоротити навантаження на апаратну частину комп'ютера, що в свою чергу скоротить вартість системи ВР [20].

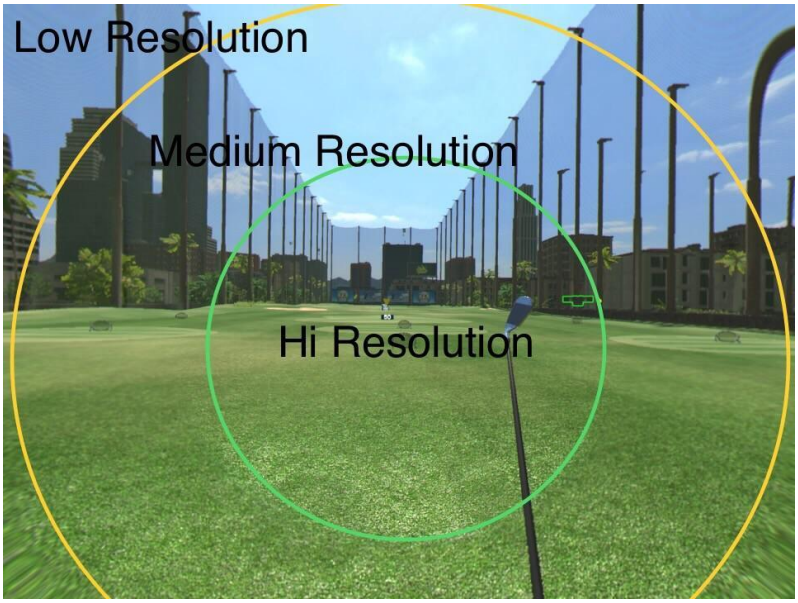


Рисунок 1.7. Процес роботи Foveated rendering [21].

Також, важливою є тема об'ємного звуку, яка активно зараз розвивається, так і інші системи, в яких задіяні додаткові відчуття. Наприклад, тактильні дії, контролери зі зворотним зв'язком, різні жилети для повного занурення тощо. Але здебільшого на поточний момент все це досить такі односпрямовані рішення, які складно інтерполювати для масового ринку. У розробників платформ є мета охопити весь комплекс відчуттів користувача.

ВР дає настільки унікальний досвід, який не можна отримати ніякими іншими способами. Також, на ринку віртуальної реальності існують killer apps.

Killer app - це додаток, сервіс, система або гра заради якої користувач готовий купити саму платформу (тобто окуляри, контролери та іншу гарнітуру віртуальної реальності) [22]. На появу killer apps для окулярів ВР чекають як самі користувачі так і виробники контенту. Багато хто вважає, що соціальні взаємодії з ВР, використання технології машинного навчання багато в чому по сприяє появі killer apps.

Це далеко не весь список, але ці перераховані вектори руху здаються найбільш актуальними в найближчій перспективі, щодо тих цілей, які я описував на самому початку. Знання цих векторів є дуже необхідною, як для розробників так і для самих користувачів ВР.

1.4 ВР на світовому ринку

Велика кількість компаній вже працює на ринку ВР. У кожної компанії є чіткий напрямок. Якісь напрямки вже опрацьовані, а якісь тільки починають розвиватися, але в цілому є розуміння того, що ринок з кожним роком активно росте та розвивається.

Всі найбільші світові компанії так чи інакше почали взаємодіяти з ВР. Хтось вкладає туди багато грошей на розвиток даної технології. Хтось веде розробку всередині ринку. Якщо брати і виділяти найбільш активні компанії, то це Facebook (Oculus) та Google (Google Glass, Google Cardboard, Google Daydream). Дані компанії інвестували туди близько кілька мільярдів доларів. Facebook підтримує відео у 360 градусів, які можна моментально інтегрувати на сторінку в соціальній мережі. Google також дав підтримку роликів 360 градусів на своєму відеохостингу YouTube. Тобто, будь-хто може завантажити відео 360 на YouTube у відмінній якості і поділитися своїм досвідом з іншими людьми. Завдяки цьому зростає досвід взаємодії ВР з людьми і ринок ВР тільки зростає [23][24].

Найбільші популярні компанії на ринку по hardware пристроях це HTC (Vive), Facebook (Oculus), PS VR від компанії SONY Entertainment та Valve Index (від американської компанії Valve). Слідом йде сумісного зі своїм рішенням компанія Samsung (Samsung Gear VR), продано кілька мільйонів пристроїв. Окремо варто виділити компанію Microsoft, вони поки не мають рішення під hardware - пристрої, але мають пристрій для змішаної реальності, це їх окуляри Hololens та Hololens 2. Ці пристрої використовують різні компанії світу як у промисловості, так і в рекламі [25][26].

Рисунок 1.8. Взаємодія користувача з окулярами HoloLens 2 [27].

У світі працює величезна кількість компаній у напрямку ВР, це і прототипна розробка, і виробництво ігор під десктоп, під мобільні рішення і під hardware. Існують також парки віртуальної реальності, де люди можуть розважатися, прийти зі своєю сім'єю і насолодитися ВР. Ринок ВР уже поступово заповнюється, і є рішення, які спеціалізуються виключно на промисловості. Практично в кожній з галузей існують кейси для створення проектів ВР.

1.5 Відмінність додатків для ПК і для окулярів ВР

При роботі з класичним персональним комп'ютером люди працюють зі звичайним дисплеєм, а віртуальне середовище знаходиться за межами екрану, тобто користувач сприймає її через рамку монітору. При цьому користувач знаходиться в реальному світі, тобто користувач прекрасно усвідомлює, що його оточує навколо, а екран є так званою четвертою стіною, що відокремлює користувача від зображення.

При роботі з ВР користувач відчуває себе всередині віртуального простору. Він не дивиться на зображення, а по суті, знаходиться в ньому, створюється так званий ефект присутності або занурення (імерсивність), саме ось це занурення досягається завдяки абсолютно іншого підходу введення і виведення інформації - це самі окуляри, які користувачі одягають на голову, це

спеціальні motion controllers, які користувачі тримають в своїх руках, які визначаються у просторі за допомогою системи трекінгу, розміщених в реальній кімнаті користувача. І всі ці системи введення і виведення інформації вимагають зовсім іншого підходу до управління контентом.

Наприклад, тепер тіло людини є частиною системи взаємодії, враховується при управлінні віртуальним середовищем жести рук, рота і голови, нахили самого тіла і так далі. Взаємодія в VR природніша, ніж з комп’ютерним додатком, так як при управлінні враховуються рухи користувача.

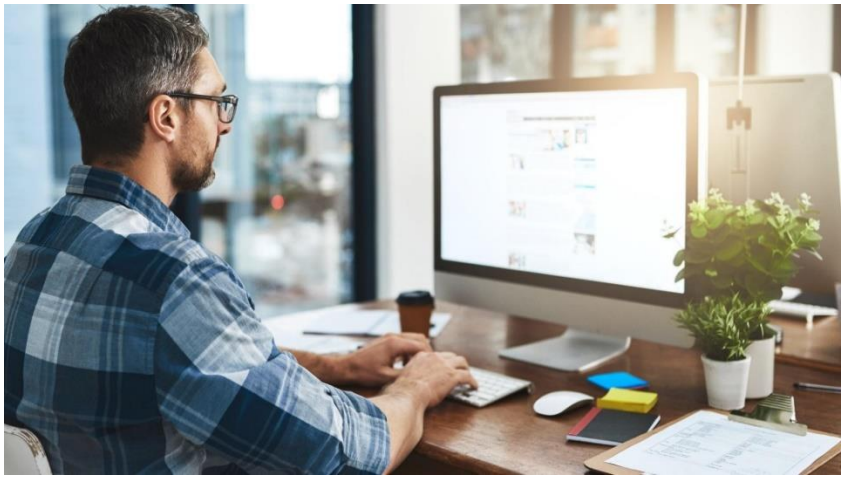


Рисунок 1.9. Взаємодія користувача з комп’ютерним додатком [28].

Наприклад, в грі від першої особи на комп'ютері за допомогою мишки потрібно навести погляд на стіл, де лежить ключ, а потім натиснути кнопку мишки, щоб цей ключ підібрати. Після чого за допомогою кнопок на клавіатурі потрібно перемістити віртуального персонажа у будинку, далі потрібно натиснути кнопку, щоб відкрити віртуальні двері за допомогою цього ключа. Але коли користувач перебуває в середині віртуальної реальності, він сам дивиться на стіл, повертає свою голову, поглядом шукає цей ключ. Далі він простягає руку за допомогою моушен контролерів, якщо вони передбачені платформою, до цього ключа, щоб підняти його, бере його і підходить до віртуальних дверей та доторкається до ручки, відкриває самі двері за допомогою підібраного ключа.

Рисунок 1.10. Взаємодія користувача з додатком ВР [29].

На рис. 1.10 зовсім інший досвід взаємодії в даному випадку з ВР. Цей досвід є більш зрозумілий користувачам, тому що використовується цілком звичні жести, рухи з реального життя. Система ВР є більш зрозумілою для користувачів, ніж звичний всім досвід роботи з персональним комп’ютером через екран монітору, клавіатури та миші.

1.6 Що таке імерсивність та для чого необхідне?

Імерсивність (від анг. слова immersion) - це відчуття себе як частини віртуального світу.

Імерсивність або ефект занурення в ВР є одним з головних особливостей системи, це відчуття присутності в всередині віртуального середовища, відключення від реального фізичного світу. Хоча визначення “імерсивність” може трактуватися досить широко, в даному випадку я маю на увазі те, що відчуває користувач, коли він стає частиною системи, що моделюється [2] [30].

Рівень імерсивності залежить від цілого ряду чинників, починаючи з апаратної частини самого обладнання, так і самого контенту. Наприклад, це якість відображення реальності, тобто наскільки великим є дозвіл дисплею, наскільки широкий кут огляду, крім цього, дуже важливий аспект синхронізації руху користувача в реальності з віртуальною камерою та контролерами.

Звичайно ж, дуже важливим є рівень графіки, не обов'язково потрібно створювати фото реалістичні об’єкти та навколишній світ, але віртуальна

реальність повинна бути приємною та її використання має бути комфортним для користувачів. У віртуальному середовищі є дуже важливим створення природних факторів, зворотних відгуків та взаємодія. То, яким чином користувач бере об'єкти за допомогою motion controllers, піднімає їх, перетягує, ставить на віртуальне поверхню та інше. Коли мозок користувача приблизно розуміє, що очікувати від того простору в якому він опинився.

Не можна не згадати об'ємний звук, який таким же чином впливає на ефект занурення, сюди ж відноситься і можливість передача тактильних відчуттів, запахів, тобто взаємодії додаткових органів почуттів людини. І чим більше задіються кількість органів почуттів під час моделювання віртуального світу, тим більше навколишній віртуальний світ стає достовірним самому користувачеві.

Звичайно, імерсивність стосується не тільки систем ВР, вона може стосуватися звичайних комп'ютерних ігор, це так звана тактичне занурення або взагалі до процесу читання книг, тобто розподільне занурення. Але саме ВР дозволяє досягти небувалого до цього часу сенсорного, просторового і психологічного занурення в зовсім інше віртуальне середовище. Створюється потужне відчуття присутності абсолютно іншого світу.

Отже, для занурення у ВР користувач використовує необхідне обладнання, віртуальні окуляри, motion controllers, систему трекінгу. І всі ці аспекти потрібно дуже гарно проробити аби у людини не почала паморочиться голова, аби її не почало нудити, і це все впливає на загальний ефект імерсивності. Тому імерсивність саме той аспект, якому повинні приділяти увагу при розробці проектів під окуляри ВР [31].

1.7 Пристрій окуляри VR та різниця між окулярами VR

Google Cardboard – це найпростіший ВР пристрій, призначений для мобільних телефонів. Він складається з картонки та лінз у середині. Лінзи необхідні, тому що без них екран телефону буде відразу перед очима

користувача і через це картинка буде розмитою, нечіткою і складно користувачеві буде сфокусуватися на ній, а лінзи вирішують цю проблему. Вони віддаляють екран від користувача на комфортну для нього відстань. Ця відстань на різних пристроях різна, але у середньому цей показник 1.5 метра. Майже усі сучасні телефони можуть працювати з Google Cardboard [32].

Рисунок 1.11. Приклад Google Cardboard [33].

Більш просунутий варіант окулярів ВР це Samsung Gear VR. Його переваги над Google Cardboard у тому, що у ньому більш щільна інтеграція разом з мобільними телефонами.

На корпусі самих окулярів є спеціальні кнопки, які ми можемо натискати для взаємодії з інтерактивними об’єктами. У Google Cardboard для взаємодії над об’єктами ВР необхідно навести погляд точкою-прицілом, яка знаходиться у користувача перед очима, на інтерактивний об’єкт взаємодії та затримати на ньому свій погляд на певний проміжок часу (зазвичай не більше декількох секунд).

Рисунок 1.12. Приклад окулярів Samsung Gear VR [34].

У Samsung Gear VR ми можемо взаємодіяти з об’єктами, як за допомогою погляду, так і за допомогою спеціальних кнопок на поверхні самих окулярів. Також є спеціальна кнопка скролінгу, для фіксації розташування лінз. Проте Samsung Gear VR більше підходить для телефонів лінійки Galaxy, а не для інших [35].

Наступний більш просунутий варіант окулярів це Oculus Rift (Facebook). В ньому вже не використовується телефон. Він має дроти, тобто це вже не бездротовий варіанті як Google Cardboard та Samsung Gear VR. Дроти у Oculus Rift необхідні для підключення до комп’ютера, причому комп’ютер повинен бути потужним, тому що всі дії для відображення готової картинки додатку у шоломі буде вираховувати саме комп’ютер, а не телефони у Cardboard чи Samsung Gear VR.

Цей шолом також має лінзи, які регулюються, а також навушники для об’ємного звуку, для більш глибокого занурення користувача у ВР.

Рисунок 1.13. Окуляри ВР Oculus Rift [36].

Також, важливою відмінною Oculus Rift від окулярів Google Cardboard чи Samsung Gear VR є те що, користувач може рухатись у реальному просторі і він так же почне рухатись у віртуальному просторі, це робиться завдяки спеціальній камері, яка стоїть на столі користувача та відслідковує його дії. У окулярах Cardboard можна рухати лише головою. І якщо в них користувач зробить крок у реальному просторі, то у віртуальному нічого не зміниться, вся

віртуальна реальність переміститься разом із ним. Тому, Oculus Rift надає користувачам повноцінний досвід для повного занурення у віртуальне середовище [37].

Наступний вид окулярів це HTC Vive. Він як Oculus Rift також має дроти, лінзи, спеціальне колесо для регулювання відстані між лінзами та підключається до комп'ютера. Також, він використовує більш розвинуту систему трекінгу (tracking) для взаємодії з об'єктами ВР [38].

Рисунок 1.14. Окуляри ВР HTC Vive [39].

Наступний вид окулярів це окуляри PlayStation VR від компанії Sony Entertainment. Цей шолом також має дроти, лінзи, спеціальне колесо для регулювання відстані між лінзами, проте вже підключається до ігрової консолі, шляхом підключення дротів до неї. Весь процес рендерінгу (rendering) відбувається всередині консолі. Для дій з об'єктами має спеціальні світлодіодні контролери. А також, має свою систему трекінгу [40].

Рисунок 1.15. Окуляри ВР PlayStation VR [41].

Окуляри ВР Valve Index є одними з останніх окулярів ВР для комп’ютерів.

Рисунок 1.16. Окуляри ВР Valve Index [42].

Ці окуляри також мають лінзи, мають вбудовані навушники, регулятори для налаштування зображення (відстань між екраном і очима користувача і кут поля зору), кабель для підключення до комп’ютеру і не можуть працювати без нього. Бездротового варіанту окулярів Valve Index поки що не існує [43].

1.8 Взаємодія і системи трекінгу пристроїв

З ВР можна взаємодіяти за допомогою погляду. Тобто, куди направлений погляд користувача там і відбувається певна дія. Або можливо взаємодіяти за допомогою контролерів (motion controllers або інші відповідні джойстики).

Взаємодія за допомого погляду є максимально простою, у користувача посеред екрану є спеціальна точка, яка виступає у якості курсору.

Наприклад, коли користувач наводить погляд на певний об’єкт для взаємодії з ним, то відбувається певна дія, або триває заповнення Progress Bar, і коли він заповнився, тоді відбувається певна дія, або коли користувач відвів погляд від об’єкту взаємодії, тоді також відбудеться певна дія, існують і інші варіанти. Так працює у Cardboard [32].

У Samsung Gear VR є спеціальні додаткові кнопки на корпусі окулярів для цього. Наприклад, можна навести погляд на об’єкт взаємодії і не чекати, коли

заповниться Progress Bar, а просто натиснути клавішу на шоломі і відбудеться запланована дія [35].

Рисунок 1.17. Наведення точки курсору на інтерактивний об’єкт [44].

Oculus Rift (Facebook), HTC Vive, PlayStation VR (Sony Entertainment) мають не лише сам шолом, а ще і спеціальні контролери для взаємодії з об’єктами всередині віртуального простору.

Система трекінгу у різних окулярах віртуальної реальності своя.

Так Google Cardboard відстежує лише напрямок, в який дивиться користувач, кути нахилу голови. Тобто, коли користувач рухає головою, то він бачить як змінюється простір, але коли він зробить крок вперед, то змін не відбудеться, весь віртуальний світ відсунеться разом із ним. Це так працює через вбудований в мобільний телефон акселерометр, гіроскоп та магнітометр. Разом вони дозволяють відстежувати нахили голови користувача, а також щоб не було смикання. У Samsung Gear VR, все працює так само, але у новій версії також є спеціальний controller, який також використовується всередині віртуального простору. Він також враховує кути нахилу, а не саме переміщення.

У Oculus Rift, HTC Vive і у PlayStation VR система трекінгу більш повноцінна, у них відстежуються положення, як самих пристроїв, так і їх контролерів у просторі, тобто користувач може переміщатися вліво чи вправо у реальному світі, і ці ж зміни будуть відбуватися у віртуальному світі також.

У Oculus Rift на панелі розташовані інфрачервоні світлодіоди, на столі у користувачів стоять спеціальні камери, які спрямовані на користувача і записують картинку, як зліва, так і справа в інфрачервоному діапазоні, також на контролерах теж є ці точки. Через тріангуляцію ми можемо знати, де дійсно знаходяться об'єкти в просторі. Схожа методика використовується для motion capture фільм. Її недолік в тому, що сама камера, яка дивиться на шолом повинна бути приєднана до комп'ютера і комп'ютер обробляє те, де знаходиться шолом. Відповідно не має можливості помістити два шолома, які отримують картинку від різних комп'ютерів в цю систему трекінгу. Тобто для кожної система повинна бути своя пара трекінгу, які приєднані до комп'ютера звідки бере картинку шолом [36].

У випадку з HTC Vive, система трекінгу пройшла трошки вперед з технологією лайтхаусес (система lighthouse). Це технологія трекінгу, коли в кутах кімнати прикріплені спеціальні коробочки, які синхронізовані між самими собою. Якщо вони бачать один одного, то синхронізація відбувається автоматично, якщо вони погано бачать один одного, то ми можемо їх синхронізувати через додатковий кабель, який пропонується в комплекті цих коробочок. Ці коробочки випускають спеціальний сигнал, який засвічує все приміщення горизонтально та вертикально в інфрачервоному діапазоні. Користувач їх не бачить, проте їх бачать и розуміють самі пристрої, шолом та контролери. Шолом знаходиться у просторі трекінгу, всередині себе має мікросхему, яка сприймає інформацію від цих lighthouses. На самому шоломі є фотоелементи і ці фотоелементи засвічуються інфочервоним кольором по одній, що дає змогу системі розуміти куди дивиться людина. А також це засвічення зменшується чи збільшується в залежності, де знаходиться людина, і це дає змогу розуміти, як вона переміщається у реальному просторі [38].

У Playstation VR працює більш проста система трекінгу. Використовується PlayStation Eye, це звичайна камера, яка знімає у RGB діапазоні, використовуються контролери на яких є білі шари, які в певний

проміжок часу загораються червоним, синім, зеленим чи фіолетовим кольорами. Через те, що самі шари матові, то вони мають суцільний колір. На самому шоломі теж присутній спеціальний фрагмент, який в спеціальний момент починає світитися певним кольором, зазвичай синім. І завдяки цим кольорам система трекінгу працює. Камера бачить, яким кольором зараз світяться контролери і сам шолом, і цю інформацію з координатами у просторі передає далі на консоль, яка в свою чергу рендерить необхідне зображення і показує його користувачеві [40].

У Index Valve система трекінгу схожа з системою трекінгу платформи HTC Vive, бо компанія Valve також використовує свою систему lighthouse. На відміну від контролерів Oculus Rift або HTC Vive, у Index Valve користувач не стискає ці контролери в руці, а кріпить до долоні і просмикує пальці в спеціальну рамку з 87 датчиками, що зчитують рухи пальців - це дозволяє пробляти складні жести, і маніпуляції з віртуальними об'єктами. Контролери Index також сумісні з шоломами HTC Vive і Vive Pro [43].

1.9 Існуючі системи віртуальної реальності та їх аналіз

У таблиці 1.1 містяться існуючі системи віртуальної реальності для допомоги людям у різних сферах життя.

У цій таблиці під “Virtual prototyping system” я маю на увазі віртуальні системи прототипування, такі системи використовуються у промисловості. Наприклад, щоб фахівці заводу могли потренуватися на тяжкому приладі у ВР, а не стразу на справжньому приладі. Кожен завод чи компанія створюють такі системи під свої власні прилади, тому такі системи не є масово доступними, а також вони достатньо дорогі для розробки [45].

Наступна у таблиці система “The Body VR”, в цій системі можна здійснювати подорож по тілу людини. Ця система допомагає у навчанні для медиків. Вона дає можливість краще засвоїти студентам матеріал будови тіла людини, тому що студенти можуть більш детально розглянути органи людини з середини. Ця система не є масово доступною але з ціновою доступністю,

оскільки її можна знайти і завантажити лише для комп'ютерної віртуальної реальності та для Samsung Gear VR цей додаток є безкоштовним [46].

Таблиця 1.1

Таблиця порівняння існуючих систем віртуальної реальності

Назва системи / Перевага	Virtual prototyping system	The Body VR	Tilt Brush	Sea Quest Hero	Other VR systems for detecting Alzheimer's	Other VR Maze System (Games)	My VR System
Масова доступність	-	-	-	-	-	+	+
Цінова доступність	-	+	+	+	-	+	+
Допомога користувачу системи	+	+	+	+	+	-	+
Допомога хворим	-	-	-	+	+	-	+
Простота (легкість у використанні)	-	+	-	-	-	+	+
Розвага	-	+	+	+	-	+	+
Тренування пам'яті	-	-	-	-	-	-	+
Перевірка орієнтація у просторі	-	-	-	+	+	+	+
Повне занурення	+	+	+	-	+	-	-

Система "Tilt Brush" – це додаток від компанії Google, в якому користувачі використовують джойстики, перебуваючи в тривимірному просторі. Користувачі можуть малювати в обсязі, використовуючи анімовані

кисті. І у цій системі є можливість імпортувати ці намальовані користувачами малюнки в форматі .fbx для подальшої роботи з ними в різних редакторах, ігрових середовищах (наприклад, Unity чи Unreal Engine). Ця система не є масово доступною, тому що вона створена виключно для стаціонарного VR. А також, ця система без цінової доступності, оскільки цей додаток платний [47].

Систему “Sea Quest Hero” використовують для визначення хвороби Альцгеймера. У цій системі користувачу необхідно постійно орієнтуватися у зануреному віртуальному просторі. Ця система не є масово доступною, оскільки вона доступна для Samsung Gear VR та Oculus, а це достатньо дорогі платформи [48].

У цій таблиці під “Other VR Maze System (Games)” я маю на увазі класичні ігрові лабіринти ВР, які доступні у різних маркетах Android чи IOS додатків. Такі системи користувачі використовують для розваги, як і будь-які інші ігри, такі системи не допомагають користувачам чи хворим людям, проте вони є масово доступні та з ціновою доступністю.

У таблиці під “Other VR systems for detecting Alzheimer’s” я маю на увазі інші системи виявлення хвороби Альцгеймера шляхом перевірки просторової орієнтації користувача. Також, до таких систем належить створений у Німеччині лабіринт віртуальної реальності, який показав, що люди, схильні до хвороби Альцгеймера, погано пройшли лабіринт на відміну від здорових людей [49], тому що при хворобі Альцгеймера певна частина мозку, яка відповідає за просторову орієнтацію, порушується [50]. Інформації про цей лабіринт мало і він недоступний для великої кількості людей. Тому є необхідність у створенні системи віртуальної реальності для визначення схильності до хвороби Альцгеймера для мобільних пристроїв, бо у такому випадку така система буде максимально доступною для великої кількості користувачів.

Як можемо бачити із таблиці порівняння існуючих систем, що моя система має ряд переваг:

1. Є масово доступною, а також із ціновою доступністю, тобто будь-який користувач смартфону з операційною системою Android та з будь-якими віртуальними окулярами може випробувати мою систему.

2. Допомагає виявляти чи має людина проблеми із пам’яттю та орієнтацією у навколишньому середовищі, тобто чи може мати користувач даної системи схильність до хвороби Альцгеймера.

3. Для запобігання зменшення прогресування хвороби Альцгеймера має можливість для тренування пам’яті.

4. Є простою у використані, користувач легко зможе розібратися у її використані, оскільки вся взаємодія з інтерактивними об’єктами відбувається за допомогою наведення погляду, тобто за допомогою нахилу голови користувача, без використання інших частин тіла та без пересування у реальному просторі.

1.10 Розгляд хвороби Альцгеймера

Хвороба Альцгеймера - є найпоширенішою причиною деменції у людей похилого та старечого віку - вона діагностується у 50% пацієнтів з деменцією. Порушення пам’яті - перша і найзначніша скарга [1].

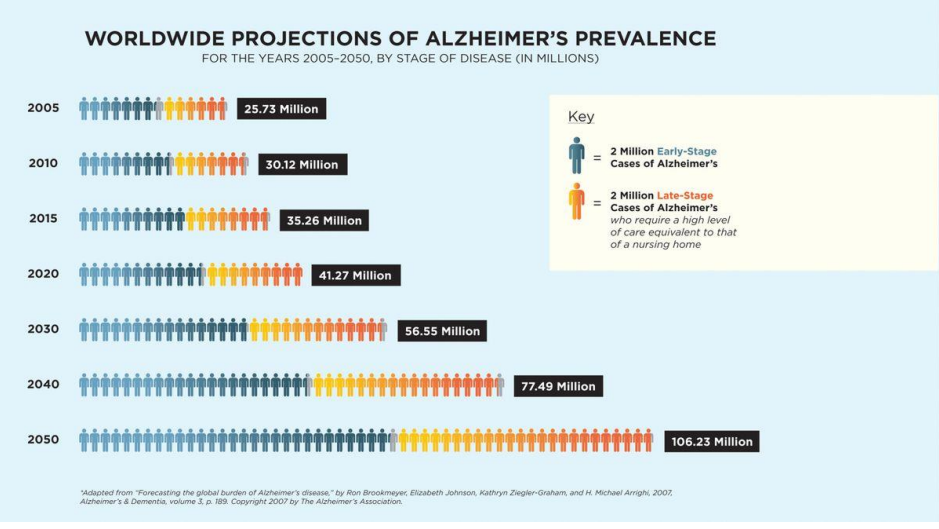


Рисунок 1.19. Світові прогнози поширеності хвороби Альцгеймера [51].

На рисунок 1.19 ми можемо побачити графік статистики щодо майбутнього прогресування хвороби Альцгеймера.

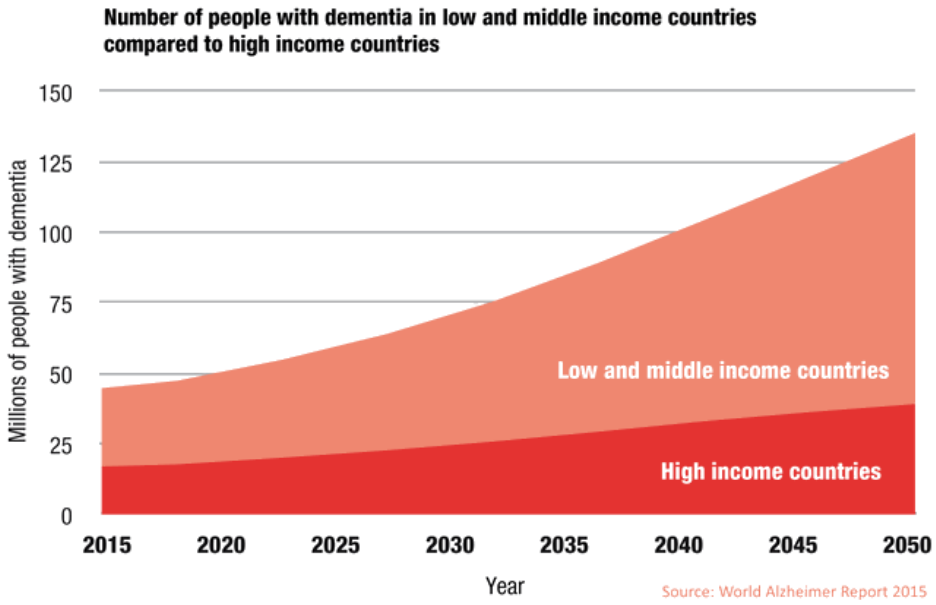


Рисунок 1.20. Кількість людей з деменцією в різних країнах [52].

На рисунку 1.20 ми також можемо побачити, що кількість пацієнтів із хворобою Альцгеймера лише зростатиме в різних країнах світу.

Також, цей рисунок показує, що деменцією страждають в основному люди похилого віку, але кількість випадків, які починаються до 65 років швидко зростає.

Отже, ці графіки показують, що хвороба Альцгеймера зараз швидко прогресує і надалі буде прогресувати. Ця хвороба зустрічається не тільки у літніх людей (старше 65 років), але і у молодших людей. Тому в наш час є велика потреба почати виявляти це захворювання у людей з раннього віку.

Деякі з основних симптомів хвороби Альцгеймера: [1]

- Страждає короткочасна пам'ять.
- Порушена просторова орієнтація.

Дуже важливо підтримувати пам'ять у нормі. Це вимагає систематичного тренування пам’яті, особливо для людей, схильних до хвороби Альцгеймера.

На основі перерахованої інформації, в даному дипломному проекті розроблено систему віртуальної реальності для визначення дій людини у лабіринті.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі були розглянуті види реальності, а точніше віртуальна, доповнена та змішана реальність. Були розглянуті їх відмінності. Була розглянута історія ВР і напрямки руху ВР, її прогнози, цілі та моменти для вирішення цих цілей.

Були розглянуті необхідні складові ВР, а саме tracking, motion controllers, eye tracking, в чому їх важливість, та як вони допомагають зробити занурення у віртуальний простір для користувача більш реальним та правдоподібним. Була розглянута складова ВР foveated rendering та яким чином вона допомагає підвищити продуктивність додатків ВР.

Були розглянуті деякі з найбільших компаній, які взаємодіють з ВР та були розглянуті існуючі платформи ВР. Були розглянуті відмінності додатків для персональних комп'ютерів і для окулярів ВР. Була розглянута імерсивність та її значення у віртуальному світі та чинники від яких вона залежить.

Були розглянуті пристрої ВР, з чого вони складаються і для чого потрібні, були розглянуті різниці між сучасними окулярами ВР. Було розглянуто як з цими пристроями взаємодіяти та їх системи трекінгу.

Було зроблено порівняння існуючих систем ВР з моєю система, також було наведено ряд переваг моєї системи.

Був зроблений огляд хвороби Альцгеймера, її прогресування та як віртуальна реальність допомагає боротися з цією хворобою. Також, було розглянуто важливість створення системи віртуальної реальності для визначення цієї хвороби.

Таким чином актуальною є задача розробки системи віртуальної реальності для визначення дій людини у лабіринті.

РОЗДІЛ 2
ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ

2.1 Стаціонарний та мобільний VR

Стаціонарні комп'ютери можуть відтворювати контент високо полігональних моделей. Також додатки для стаціонарного VR мають дуже високоякісну графіку, що не може собі дозволити мобільний VR. Всі системи стаціонарного ВР можуть відслідковувати рухи користувача у просторі, а це значить, що користувач може фізично пересуватися. Стаціонарний ВР краще погружає людину у віртуальний простір ніж мобільний VR. Гарна, якісна та чітка картинка, дуже приємний та природній об’ємний звук, контролери для простої та привичної взаємодії користувача з активними об’єктами ВР, пересування користувача у реальному просторі дають найкращий досвід користувачам, вони максимально сильно поглинають у віртуальне середовище. Проте є декілька значних мінусів стаціонарного VR на сьогоднішній день.

Одним з мінусів стаціонарного ВР є те, що у них є дроти. І коли користувач починає переміщуватись у просторі, то він легко може заплутатись у цих дротах.

Другий значний мінус стаціонарного ВР це велика ціна. Із-за великої ціни самої платформи мало хто може дозволити собі придбати її та випробувати готові системи для неї. Окрім придбання дорогого шолому та контролерів до нього, необхідно також мати достатньо потужний комп’ютер, аби мати можливість запускати додатки під ВР. Отже, стаціонарна ВР має свої переваги, а також свої недоліки.

Мобільна ВР складається з мобільного телефону і гарнітури ВР, в яку цей мобільний телефон вставляється чи, в яку цей телефон підключається. На ринку дуже велика кількість гарнітур ВР для мобільних телефонів. Їх можна розділити на фірмові та найпростіші гарнітури.

Найпростіші гарнітури найчастіше являють собою картоні варіанти, які досить дешеві, але не довговічні. Зрозуміло, що картон при постійному використанні починає псуватися і рано чи пізно вся ця конструкція розвалюється. Пластикові гарнітури довговічні, вони володіють різними системами для того, щоб сфокусувати картинку, та для того, щоб налаштувати телефон для максимального комфорту зору.

Фірмові гарнітури це зовсім інший рівень якості обладнання. Зараз на ринку є два великих представника, це Samsung Gear VR та Google Daydream. Відмінність цих гарнітур від інших в тому, що вони мають спеціальну екосистему, яка вбудована вже виробником самих цих пристроїв, які дозволяють максимально оптимізувати роботу програми ВР. Тобто запущені на цих телефонах програмне рішення буде більш оптимізоване та працювати буде більш продуктивніше.

Основним мінусом фірмових гарнітур є те, що до них підходять не всі телефони, а тільки фірмові. Наприклад, у гарнітурах Samsung Gear VR не можна підключити інші телефони окрім лінійки телефонів Samsung Galaxy.

Google Daydream це ідеологічне продовження Cardboard від компанії Google. Особливість Google Daydream в тому, що в ньому інтегрована з операційною системою Android набір тих програмних специфікацій і апаратного забезпечення, що гарантує максимальну продуктивність додатків віртуальної реальності.

У новій версії Samsung Gear VR з’явився спеціальний контролер, в якому є кнопка і touchpad для взаємодії з віртуальним світом. Його особливість в тому, що його можна повертати у просторі. Не пересовувати його, а саме повертати. Тепер непотрібно взаємодіяти з об’єктами ВР, наводячи погляд і чекати заповнення Progress Bar, це можна зробити завдяки контролера. Тобто коли курсор, який знаходиться перед очима, наводимо на необхідний об’єкт і натискаємо кнопку контролера відбувається необхідний сценарій.

Рисунок 2.1. Окуляри ВР Google Daydream [53].

Отже, мобільна ВР є максимально доступною для людей. Адже, щоб її випробувати, не потрібно тратити великі гроші на дорогі шоломи, гарнітури, потужний комп'ютер та на інше обладнання. Достатньо лише мати сучасний смартфон та недорогі окуляри ВР для мобільних телефонів. В ній немає дротів, які можуть тільки заважати користувачеві. Загалом, розробляти проект під систему ВР є гарним рішенням, якщо стоїть мета зробити цей проект максимально допустимим для великої кількості користувачів, а також для більш простого процесу взаємодії користувача з системою ВР.

2.2 Пристрої управління VR

На сьогоднішній день існує цілий спектр рішень систем управління ВР. Працюючи з ВР, головним інструментом є саме тіло користувача, тому важливим є врахування його ергономіки. Перенесення рухів реального користувача у віртуальний простір повинно бути зручне і комфортне. Існує декілька механізмів взаємодії.

Перший механізм – це управління поглядом. Відразу потрібно відзначити, що управління поглядом мається на увазі повороти нахилу голови користувача. Коли користувач помічає активний елемент, наприклад, кнопку, він дивиться на неї, затримує свій погляд на певний проміжок часу, поки не заповниться Progress Bar, після цього відбувається певна дія. І ніякі додаткові пристрої тут не потрібні для взаємодії, лише повороти головою. На деяких окулярах

присутні кнопки та touch панелі, які прискорюють процес управління контентом. Також, використовуються прості класичні ігрові контролери, за допомогою яких відбувається взаємодія.

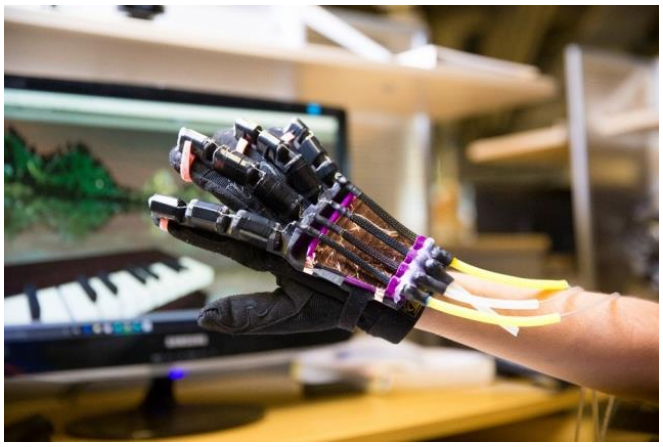


Рисунок 2.2. Рукавиці віртуальної реальності [54].

Існує система для захоплення дрібної моторики пальців рук, це спеціальні рукавиці з датчиками і коли людина згинає чи розгинає пальці у цих рукавицях, то ці дії переносяться у віртуальний простір (Рисунок 1.19). Це дозволяє створити цілком реалістичні відчуття руки у віртуальному просторі.

Рисунок 2.3. Взаємодія користувача з пристроєм Leap Motion VR [55].

Наступний розвинутим пристроєм управління VR є Leap Motion (Рис. 1.20). Він кріпиться на фронтальну панель очок. Він відслідковує рухи користувача і переносить їх у віртуальний простір, але його мінус у тому, що руки повинні знаходитись постійно перед лицем, щоб вони попадали у поле

зору самого пристрою. Це не є зовсім природньо і від цього користувачі можуть втомлюватись [56].

Також, у таких шоломах як HTC Vive, PlayStation VR чи Valve Index є свої системи трекінгу, які візуалізують контролери, які знаходяться у користувача в руках, у ВР. У цій роботі, в мене стоїть задача зробити систему ВР для великої кількості користувачів, а отже при створені системи я маю розраховувати лише на нахили голови користувача і щоб вся взаємодія з інтерактивними елементами відбувалася завдяки погляду користувача.

2.3 Unity. Що це та для чого?

Unity – це ігровий рушій для створення різних проектів для різного напрямку. Це можуть бути ігри для комп'ютерів (PC, Mac, Linux) чи мобільних пристроїв (Android/IOS), для різних видів консолей (Xbox One, PS4). Unity використовують для творення веб-додатків. Також, Unity використовують для створення проектів VR і AR. [57]

Програма Unity працює, як на OS Windows, так і на OS Mac. Потрібно відзначити, що зараз є можливість працювати в цій програмі на OS Linux.[58]

Функціональні можливості

Програма Unity має дуже зручний інтерфейс для розробників. Редактор програми складається з багатьох вікон. Самими важливими вікнами для роботи з редактором Unity є:

1. Scene;
2. Game;
3. Inspector;
4. Hierarchy;
5. Project;
6. Console;
7. Animator;
8. Animation.

Scene – це вікно, в якому розробник може пересуватися по сцені. Також, може в цьому вікні пересувати об’єкти, розташовувати їх де це потрібно, змінювати їх масштаб, повертати їх. Для цих дій у Unity є спеціальний Toolbar з декількома функціями, а саме: [59]

1) Hand Tool – для можливості розробнику переміщатися по сцені.
2) Move Tool – для можливості переміщення об’єктів сцени.
3) Rotate Tool – для можливості повертати об’єкти сцени.
4) Scale Tool – для можливості змінювати масштаб об’єктів сцени.
5) Rect Tool – для можливості змінювати пропорції об’єкту, як прямокутника.
6) Move, Rotate or Scale selected objects – це комбінована функція, для зміни розташування, повертання та масштабу об’єктів сцени. [60]

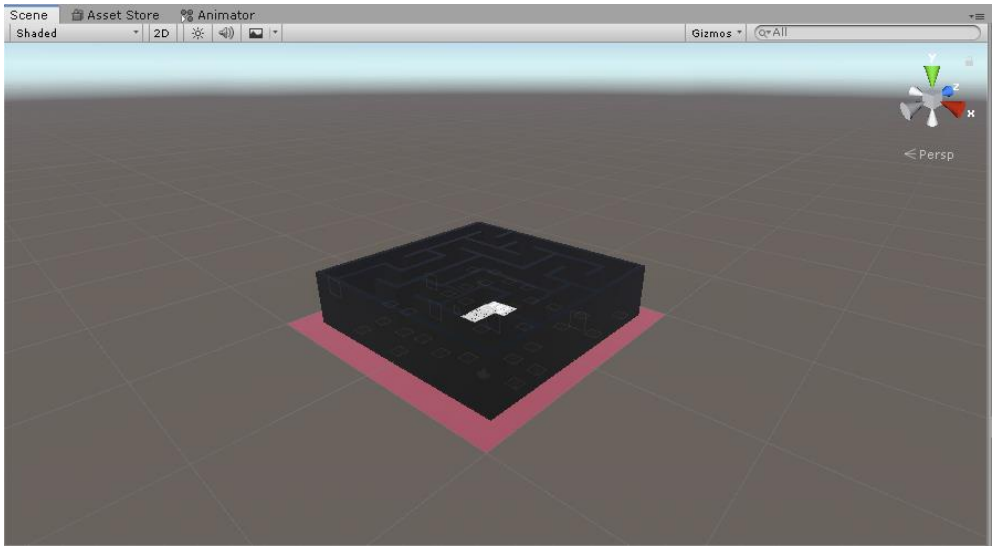


Рисунок 2.4. Вікно Scene.

Рисунок 2.5. Toolbar з функціями для роботи у редакторі Unity.

На рисунку 2.5 ми також можемо бачити, що є можливість для зміни Center на Pivot та Global на Local. Center чи Pivot використовується для зміни точки опори, це впливає на зміну позиції. Global чи Local використовують для зміни повороту відносно самого об’єкту чи світової просторової орієнтації [61].

Game – це вікно для відображення процесу гри у самому редакторі Unity. Це є дуже зручним, оскільки є можливість подивитися процес гри і пограти в розроблену гру у самому редакторі без необхідності збірки проекту на обрану ОС певного пристрою і запуску її там.

Також, у цьому вікні можна не тільки бачити весь процес гри, але і змінювати розширення екрану та положення екрану, для мобільних пристроїв воно може бути вертикальним або горизонтальним.

У вікні Game є наступні вкладки: “1920x1080 Landscape”, “Scale”, “Maximize On Play”, “Mute Audio”, “Stats”, ”Gismos”.

- Вкладка “1920x1080 Landscape” відображає розширення екрану смартфону та горизонтальне положення екрану. В цій вкладці можна вибирати різні розміри екранів та положення телефону. Це дає можливість подивитися розробнику як об’єкти додатку будуть виглядати для мобільних пристроїв з різними розширеннями екрану.
- Вкладка “Scale” дозволяє змінювати масштаб, це може знадобитися, щоб більш детально роздивитися об’єкт гри.
- Вкладка “Maximize On Play” дозволяє відкривати вікно Game у повноекранному варіанті після запуску гри у редакторі.
- Вкладка “Mute Audio” дозволяє виключити звук ігри під час перегляду процесу гри, це є дуже зручним для розробника, щоб музика не заважала під час перевірки процесу гри.
- Вкладка “Stats” дозволяє подивитися статистику гри, таку як рівень аудіо, швидкість ядра, кількість FPS (frame per second) та інше.
- Вкладка “Gismos” використовується для візуального налагодження чи налаштування у вікні "Сцена" [62].

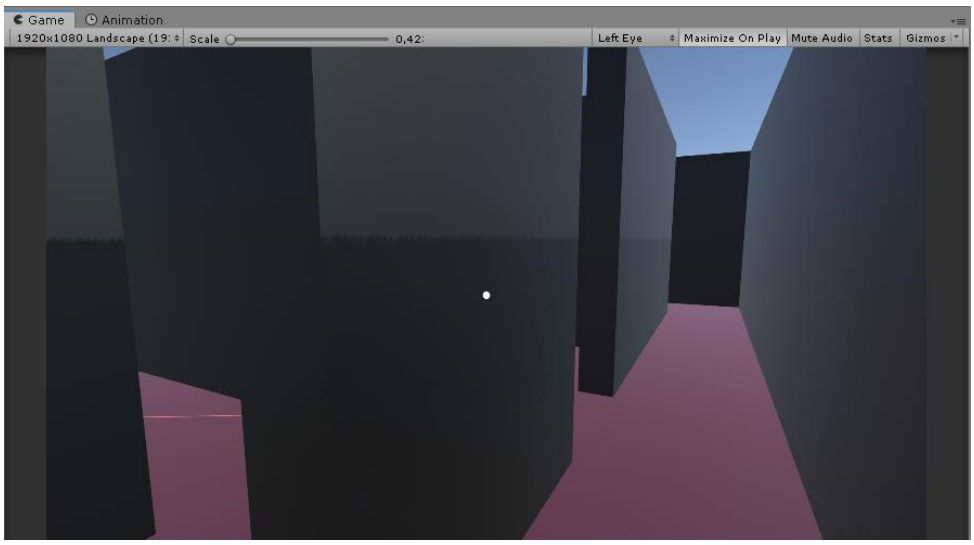


Рисунок 2.6. Вікно Game.

Для записку гри, паузи гри чи для того щоб перемотувати по кадрова гру існує друга частина Toolbar [61].

Рисунок 2.7. Частина Toolbar для запуска, паузи та перемотування гри.

Inspector – це вікно у редакторі Unity, яке показує компоненти активних об’єктів. Компонентів у Unity дуже багато і у кожного компонента своє призначення. Це вікно містить назву об’єкта, тег об’єкта і можливість робити його статичним чи ні. Тег можна використовувати для більш простої роботи з об’єктом, який цей тег містить, наприклад, для пошуку об’єкта в сцені Unity по назві тегу. Static використовується в нерухомих об’єктах сцени, це дозволяє підвищити продуктивність гри. Також, вікно Inspector містить компонент Transform.

Transform – це компонент у редакторі Unity для редагування позиції, повороту та масштабу об’єкту по всім трьом координатам, цей компонент за замовчуванням містять усі об’єкти сцени [63].

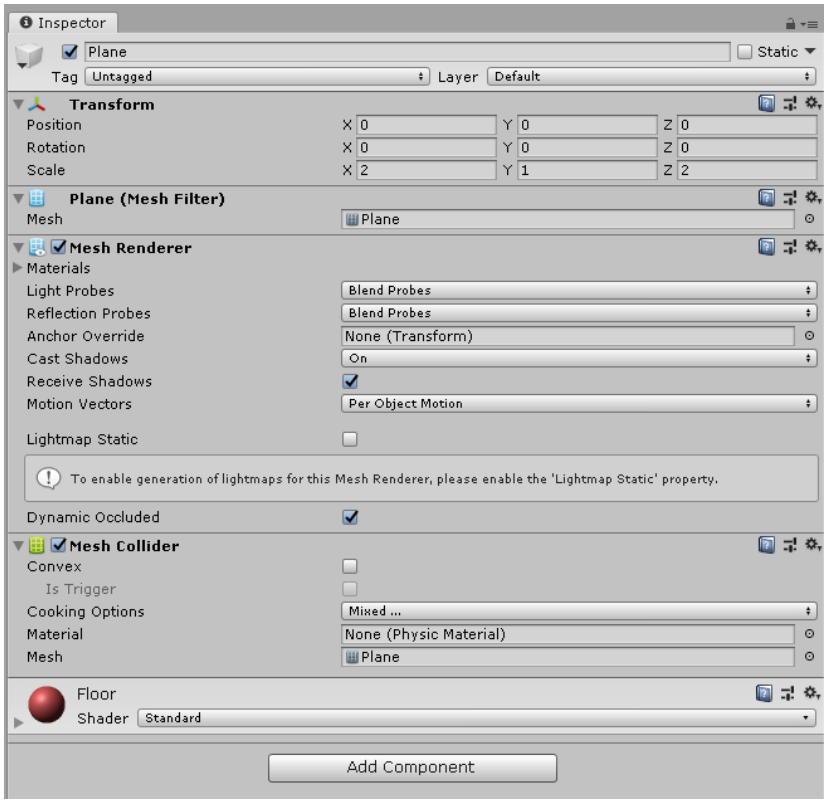


Рисунок 2.8. Вікно Inspector.

Hierarchy – це вікно у редакторі Unity, яке відображає усі об’єкти, які присутні у сцені. У цьому вікні є можливість помістити одні об’єкти у інші, тобто є можливість робити дочірніми об’єкти для родительського. У це вікно можна перетаскувати об’єкти з вікна Project або створювати об’єкти в самій сцені за допомогою вкладки “Create”. Наприклад, можна створити елементи UI [64].

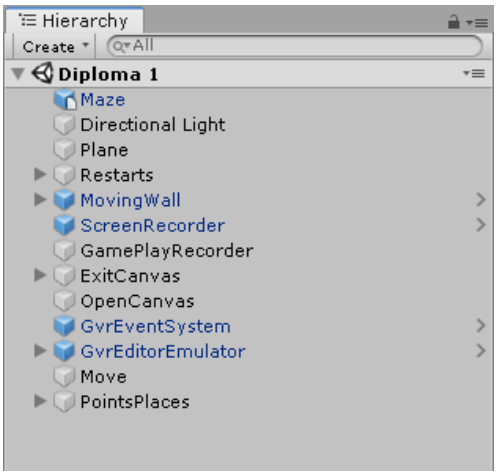


Рисунок 2.9. Вікно Hierarchy.

Project – це вікно, яке виступає як провідник з папками проекту. В цьому вікні зберігаються усі необхідні об’єкти, такі як скрипти, префаби, картинки, відео, матеріали, моделі та інше. Ці об’єкти можна перетаскувати на сцену, тобто на вкладку Hierarchy і ці об’єкти відобразяться у сцені. Також, у цьому вікні є вкладка “Create” для створення файлів, наприклад, для створення скриптів [65].

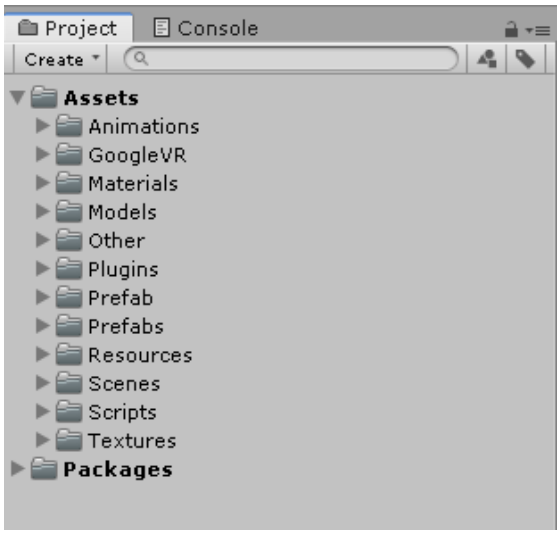


Рисунок 2.10. Вікно Project.

Console – це вікно, яке відображає помилки, які були допущені у програмному коді або відображає помилки у самому редакторі. Це вікно також відображає підказки [66].

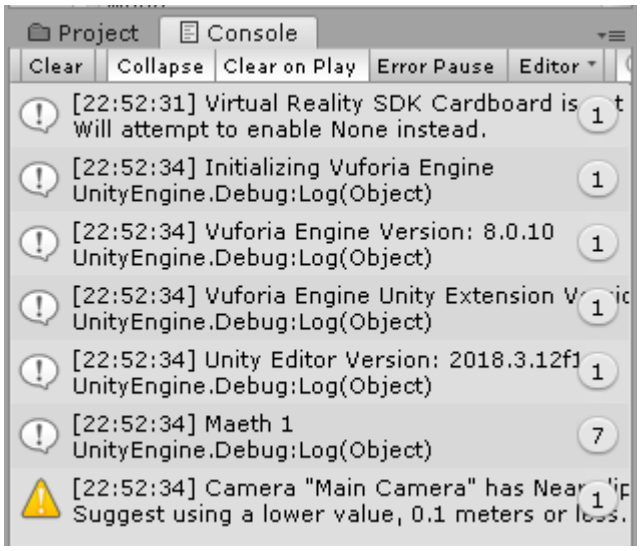


Рисунок 2.11. Вікно Console.

Animator – це вікно, в якому можна створювати зв’язки між анімаціями. В Unity можна створювати круті анімації для того, щоб оживити об’єкти у сцені гри. Також, в цьому вікні є слої та параметри, які допомагають для створення вірної взаємодії між анімаціями [67].

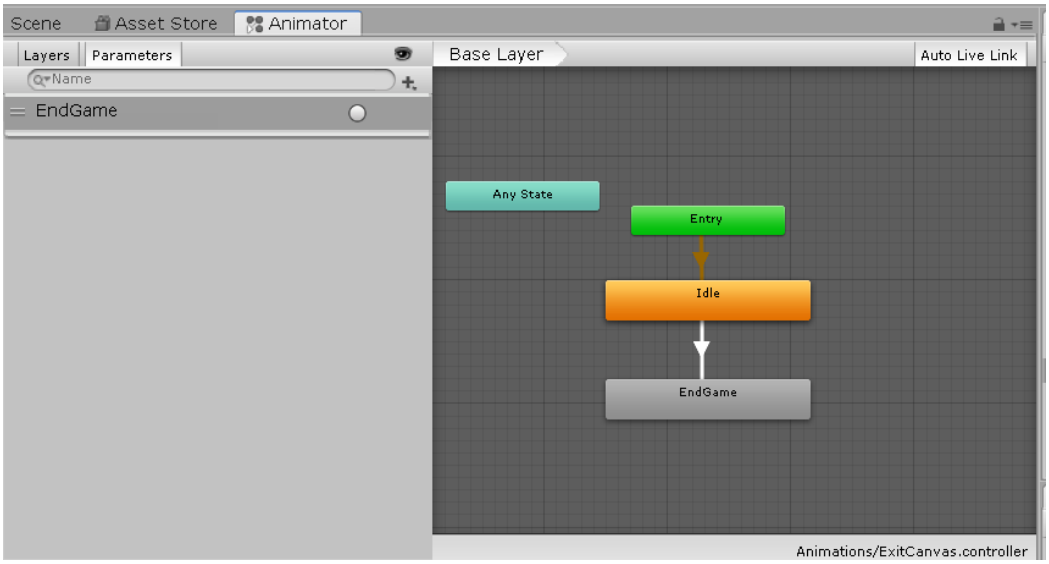


Рисунок 2.12. Вікно Animator.

Animation – це вікно, в якому можна створювати анімації для проекту. Тут є можливість встановлювати час дії анімації. У цьому вікні, наприклад, є можливість записати анімацію зміни позиції та масштабу об’єкту [68].

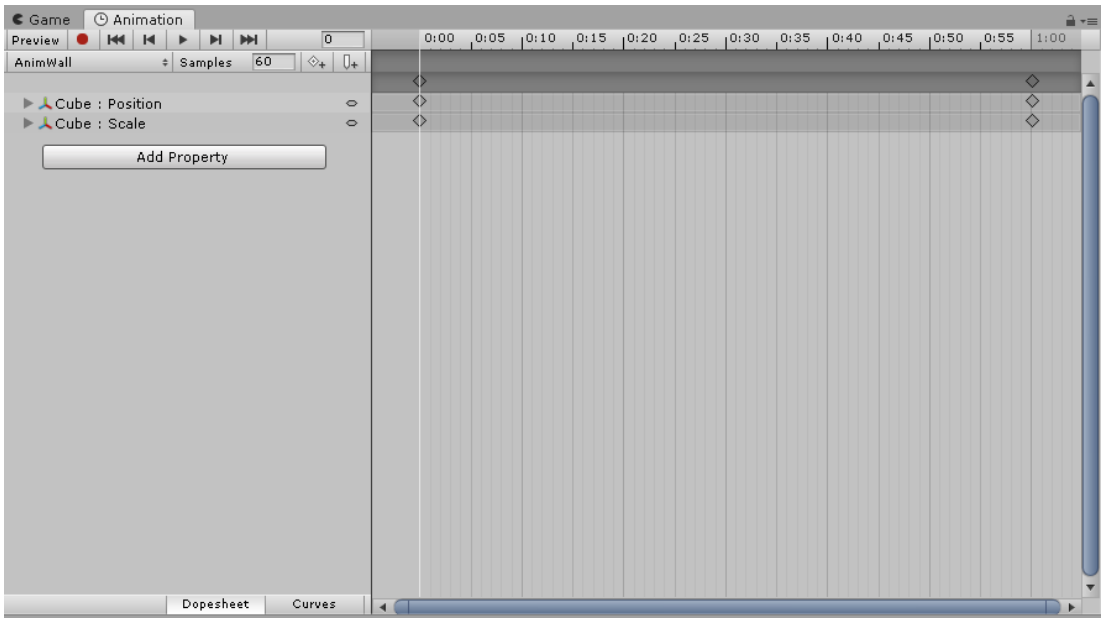


Рисунок 2.13. Вікно Animation.

Вікно Animation містить вкладку Curves для більш детальної роботи з анімаціями, там є можливість працювати з графіками рівнів руху анімації.

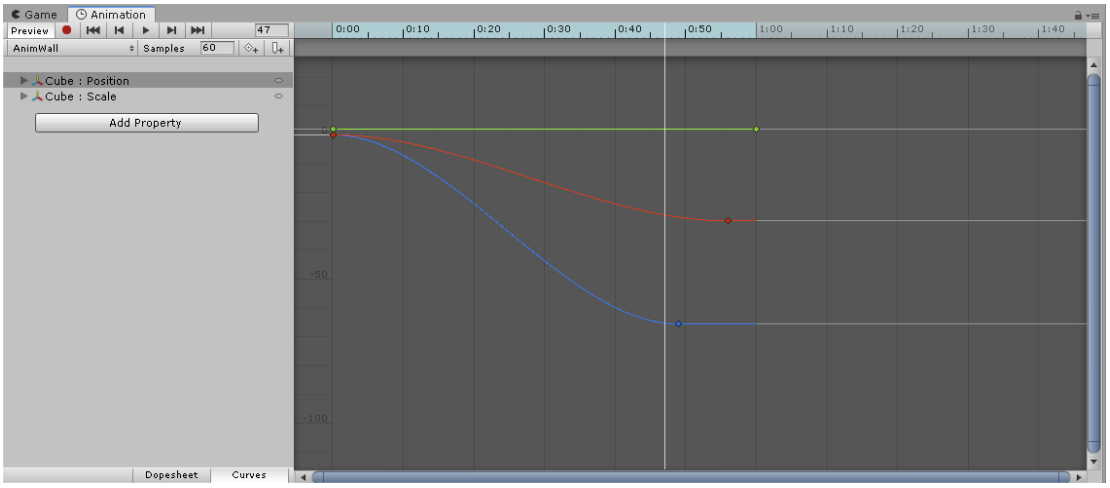


Рисунок 2.14. Вкладка Curves у вікні Animation.

На рисунку 2.14 ми можемо бачити різнокольорові лінії, які можна змінювати для налаштування більш детальної анімації яка змінюється з часом.

Технічні характеристики Unity

Unity має велику функціональність, цей ігровий рушій дає багато можливостей для розробників, щоб створювати свої проекти. Серед деяких технічних характеристик є: [69]

- У Unity є можливість розробникам створювати свої власні вкладки та вікна у редакторі;
- У Unity логіка ігор пишеться на мові програмування C# чи JavaScript. У цьому проекті я весь програмний код писав на мові програмування C#.
- У Unity є підтримка імпорту різних форматів файлів.
- У Unity можна створювати як 3D так і 2D додатки.
- У Unity можна робити дуже детальну настройку графіки, звуку та багато іншого.

Я привів лише найголовніші технічні характеристики ігрового рушія Unity.

2.4 VR та Unity

Головною платформою для розробки проектів ВР є програма Unity. Більше 60% усього віртуального контенту розроблено за допомогою Unity [70].

Unity підтримує процес рендерінгу (rendering) високого дозволу (HDRP) для VR. Це дозволяє створювати проекти ВР з високим дозволом якості відображення без зниження продуктивності.

Створення захоплюючих, фото реалістичних вражень VR за допомогою конвеєра візуалізації високої чіткості є, безумовно, дуже важливою особливістю рушія Unity [71].

Програма Unity, завдяки своїм сучасним методам візуалізації HDRP може надавати приголомшливі фото реалістичні візуальні ефекти з якістю, яке рідко зустрічалось раніше в середовищах ВР.

VR для HDRP в даний час доступний для наступних платформ і пристроїв:

- Oculus Rift & Rift S (плагін Oculus XR, Windows 10, DirectX 11);
- Windows Mixed Reality (плагін Windows XR, Windows 10, DirectX 11);
- PlayStation VR.

OpenVR: Valve в даний час розробляє свій плагін OpenVR Unity XR для 2019.3 і пізніших версій, і це буде доступно найближчим часом.

В Unity є можливість додати до своїх проектів ВР необхідні ефекти. Для цього використовується Particle System. Завдяки їй є можливість підключати різні модулі, включати форми, розміри, зіткнення, текстури і багато іншого, з легкістю додаючи потрібні ефекти.

Важливим елементом у Unity для покращення віртуального проекту є налагодження просторового звуку. Тобто є можливість забезпечити посилений ефект присутності в віртуальних середовищах завдяки вбудованій підтримці звукозапису з просторовим звучанням, полносферного об'ємного звуку,

можливостям налаштування орієнтації звукових полів відповідно до положення слухача і багато чого іншому [72].

Наступною особливістю є стерео дублювання. Це дає змогу робити контент підрисовування більш ефективнішим, використовуючи методи прискореного рендерінгу, що знижують вплив на продуктивність VR.

Також, команда Unity розробила нову архітектуру, яка відкриває нові можливості для підтримки існуючих і майбутніх платформ VR і AR реальності [72].

Найпопулярніші VR проекти створенні на платформі Unity

В світі вже існує велика кількість розроблених проектів ВР на платформі Unity. Ці проекти застосовуються і допомагають людям у різних сферах життя, незважаючи на те, що Unity є саме ігровим рушієм [72].

Найпопулярніші серед них:

1. Beat Saber

Опис: Ритмічна гра для VR, що отримала безліч нагород і поєднує в собі оригінальну електронну танцювальну музику, з любов'ю розроблені рівні та красиві віртуальні світи.

Галузь : ігри.

Розробник: Beat Games.

2. Jaguar I-PACE

Опис: Jaguar представила аудиторії свій перший повністю електричний універсал з інтерактивним віртуальним додатком. Тепер клієнти можуть познайомитися з історією дизайну і проектування концепції I-PACE в додатку, що поєднує анімацію і імерсивні VR-технології.

Галузь: автомобільна

Розробник: REWIND

3. Bonfire

Опис: Bonfire - це інтерактивний фільм у ВР, що дозволяє глядачеві взяти безпосередню участь у дії.

Галузь: медіа і розваги

Розробник: Baobab Studios

4. Додаток Walmart VR Training

Опис: Walmart розробила програми підготовки співробітників по всій країні, запровадивши VR-технології. Всі магазини в США переходять на єдину платформу навчання для мільйонів співробітників з використанням гарнітур Oculus VR.

Галузь: тренування

Розробник: Strivr

2.5 Google VR SDK

Для того, щоб мати можливість симулювати роботу ВР у редакторі Unity необхідно скачати та імпортувати asset Google VR SDK на платформу Unity. Це SDK зберігає в собі всі необхідні папки з файлами для розробки VR проектів, таких як: спеціальні префаби, моделі, текстури, плагіни, скрипти та інше [73].

Префаб (з англ. Prefab) – це вже готові об’єкти розробником, для подальшого використання у проекті і зручного їх дублювання з усіма їх компонентами. Маючи у своєму проекті багато копій (екземплярів) одного префабу, можна легко змінивши одну копію, натисканням однієї кнопки зробити заміну для всіх інших копій. Це є дуже зручним у роботі редактора Unity [74].

Головними префабами для симулювання у пакеті Google VR SDK є префаби з назвою GvrEventSystem, GvrEditorEmulator, та GvrReticlePointer.

Префаб **GvrEventSystem** необхідний для того, щоб замінити звичний об’єкт EventSystem (система подій у Unity), який використовується коли відбувається якась подія, наприклад коли кнопка натиснута користувачем. А префаб GvrEventSystem дає змогу працювати з подіями у віртуальній

реальності, коли користувач не натискає на кнопки на екрані, а наводить свій погляд на них.

Існує декілька подій, які відслідковує об’єкт GvrEventSystem, і для їх виконання необхідно зробити послідовність певних етапів:

1. Для інтерактивного об’єкту додати компонент “Event Trigger”, який має список подій. Щоб вибрати одну подію з списку подій, необхідно натиснути кнопку “Add New Event Type” та обрати необхідну подію (Рисунок 2.15). Наприклад, обравши тип події “PointerEnter”, то запрограмована дія відбудеться після того, як користувач наведе погляд на об’єкт, який має компонент з цим типом події.
2. Для того, щоб відбулася запрограмована дія, необхідно в лист “Pointer Enter” додати об’єкт, який має скрипт з написаним програмним кодом. Для того щоб його додати необхідно натиснути “+” (Рисунок 2.16).
3. Перенести об’єкт зі сцени за допомогою миші, використовуючи функцію Drag and Drop, у необхідну комірку або натиснувши на необхідний елемент (Рисунок 2.17).
4. Вибрати потрібний об’єкт в папці “Asset” чи у самій сцені (Рисунок 2.18).
5. Після того, як об’єкт з необхідним сценарієм був доданий, потрібно вибрати назву скрипта та метод, який буде відбуватися (Рисунок 2.19).

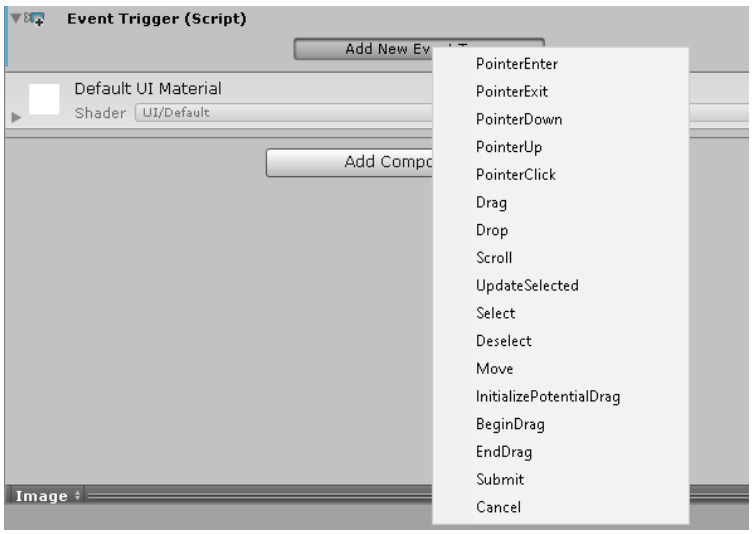


Рисунок 2.15. Типи подій компонента "Event Trigger".

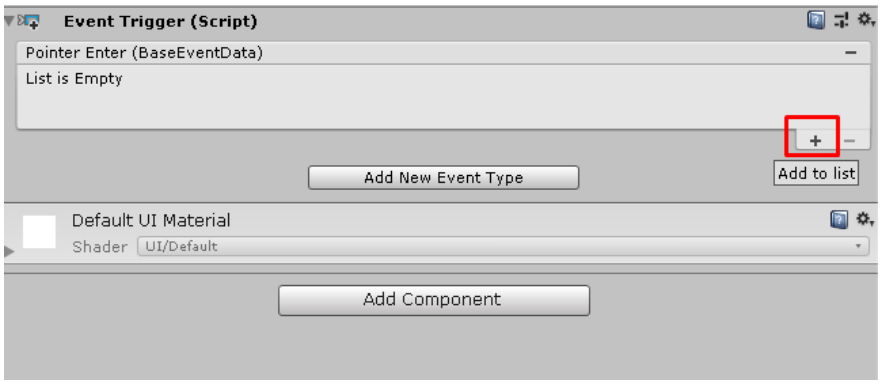


Рисунок 2.16. Додавання пустого об’єкта у список.

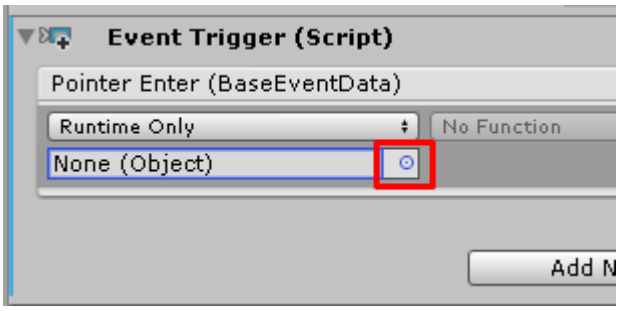


Рисунок 2.17. Елемент, на який необхідно натиснути для вибору об’єкта.

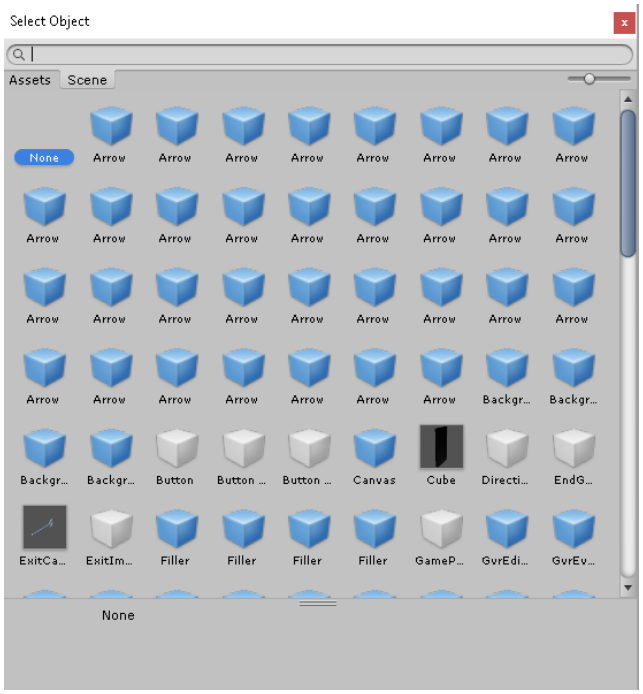


Рисунок 2.18. Об’єкти для обрання.

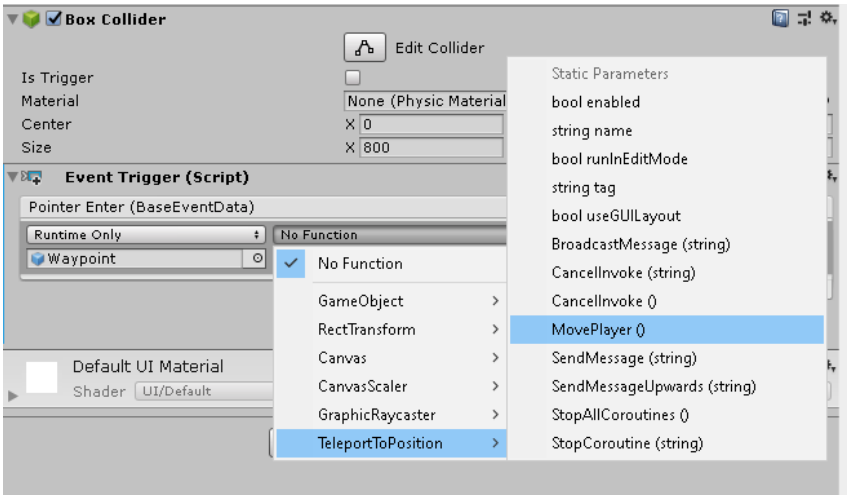


Рисунок 2.19. Вибір методу скрипта для виконання.

Префаб **GvrEditorEmulator** необхідний для того, щоб мати можливість симулювати VR окуляри у самому редакторі Unity. Це є дуже зручною опцією, бо не потрібно постійно робити збірку для Android телефону і постійно перевіряти свій додаток на ньому використовуючи окуляри ВР, а можна зробити всю перевірку у самому редакторі.

Префаб **GvrReticlePointer** необхідний для використання точки, яка виступає курсором, яка знаходиться перед очима користувача у самому

редакторі Unity. Цю точку необхідно наводити на інтерактивні об’єкти для взаємодії з ними.

Плагіни (з англ. Plugins) – це необхідні файли, які містять в собі всю необхідну конфігурацію, налаштування та параметри. Вони необхідні, наприклад, для зборки готового проекту. Наприклад, для мого проекту, це apk файл. Файл з розширенням .apk - це загрузочний файл для операційної системи Android [75].

Скрипти (з англ. Scripts) – це програмний код для роботи в Unity. Цей код пишеться для логіки гри, або, як кажуть, для сценарію гри. Зазвичай написаний на мові програмування C#, також може бути написаний на мові JavaScript (більш точніша назва UnityScript), проте ця мова більше не доступна в нових версіях Unity, тому на сьогоднішній день можна писати весь програмний код у Unity тільки на мові програмування C# [76]. У цій роботі я писав весь програмний код на мові програмування C#. Програмісти від компанії Google вже написали готові скрипти, що знаходяться у пакеті Google VR SDK, ці скрипти допомагають Unity-розробникам під час створення проектів під ВР.

Об’єкти, які мають компонент Button, являются кнопками. Тобто Unity розуміє це і дає можливість працювати з цим об’єктом як з кнопкою. Сам компонент Button має декілька параметрів та метод On Click(), який виконує написаний скрипт (програмний код) об’єкту, який ми у даний метод передаєм при натисканні кнопки. Але для роботи с системою віртуальної реальності необхідно використовувати не Button компонент, а компонент Event Trigger, даний компонент має декілька параметрів, які виконують написаний скрипт при наведені погляду або якщо ви відвели погляд, при натисканні спеціальної кнопки на шоломі та інші різні способи, більш детально я про це написав раніше.

Пакет Google VR SDK представляє собою Asset. Assets в Unity це спеціальні пакети, які містять в собі різні об’єкти для роботи в редакторі Unity. Ці пакети можна як імпортувати так і експортувати.

2.6 C#, .NET, Mono

Мова програмування C#

C# - одна з популярних мов програмування, яка існує з 2000 року (випустила компанія Microsoft) і дала можливість розробникам створювати безпечні та надійні програми, що працюють на .NET або Mono. Мова C# надає велику бібліотеку корисних класів для розробників. Мова C# реалізує всі об'єктно-орієнтовані концепції, такі як інкапсуляція, поліморфізм та успадкування, і це стає більш надійним, оскільки нові функції надаються з кожним новим оновленням [77].

Платформа .NET

.NET - платформа для розробників, яка складається з інструментів, мов програмування та бібліотек для створення багатьох різних типів додатків. Існують різні реалізації .NET. Кожна реалізація дозволяє виконувати виконання .NET-коду у різних операційних системах - Linux, macOS, Windows, iOS, Android та багато інших [77].

Платформа .NET Framework

.NET Framework - це платформа для розробки програмного забезпечення, розроблена Microsoft для роботи на платформі Windows. Програми, написані для .NET Framework, виконуються в програмному середовищі (віртуальній машині виконання), а не в апаратному середовищі. Ця віртуальна машина виклику відома як загальна мова виконання CLR (Common Language Runtime), один з головних компонентів .NET Framework, яка головним чином відповідає за управління пам’яттю, безпеку та обробку винятків [77].

Двома основними компонентами, які створюють .NET Framework, є –

1. Common Language Runtime (CLR)
2. .NET Framework class library (FCL)

FCL - це величезна бібліотека типубезпечних класів та незалежних від мови класів, які розробники використовуватимуть під час написання коду. Ці класи упорядковані відповідно до їх функціональності під назвою Namespaces.

CLR - це компонент виконання .NET Framework, який виконує функції віртуальної машини для запуску компільованого коду, створеного компілятором .NET у верхній частині ОС Windows. Компіляція коду, написаного будь-якою мовою .NET, виконується відповідними компіляторами, наприклад, для C # це компілятор C #, для VB - компілятор VB тощо. [77]

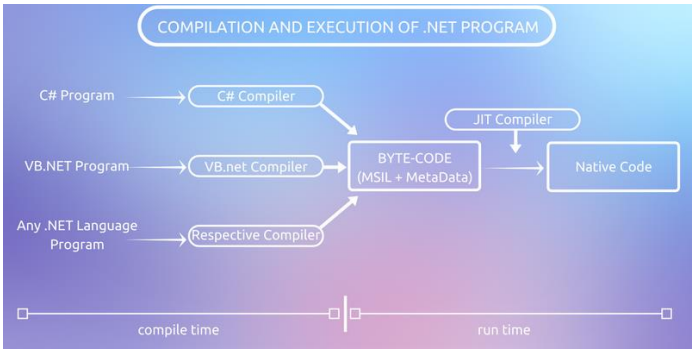


Рисунок 2.20. .NET компіляція програми [77].

На рисунку 2.20 можна ознайомитись з етапами компіляції та виконання програм, написаних за допомогою .NET Framework Language, таких як C # або VB. Детальніше про етапи компіляції:

1. Програміст пише код за допомогою C #, VB або будь-якою іншою мовою платформи .NET.
2. Під час компіляції цього коду за допомогою компілятора C# або компілятора VB.net він не безпосередньо перетворюється на бінарну/нативну мову. Замість того, щоб він перетворився на MSIL (проміжна мова Microsoft) або IL (проміжна мова) разом з деякими метаданими. MSIL (IL) + метадані, разом відомий як байт-код. Отже, отримується байт-код після компіляції. Це збірка, це те, що відомо як керований код. **Керований код** (Managed code) - це .NET-код (VB.NET, C # тощо), який розробник пише та компілює в .NET CIL.

3. Потім збірка готова до розгортання на пристроях. Також, поряд із збіркою додається загальна мова виконання (CLR), яка приєднана до збірки. Під час виконання коду компілятор Just In Time (JIT) загальної мови виконання (CLR) використовує метадані для перетворення проміжної мови (IL або MSIL) у нативний код, який також відомий як некерований код. Таким чином, байт-код (Bytecode) перетворюється в нативний (Native) код (некерований код) і після цього програма працює на Windows. **Некерований код** (Unmanaged code) - це код, який не належить до .NET, який компілюється у машинний код [77].

Mono Framework

Платформа Mono вийшла як проект з відкритим кодом у 2001 році, який приніс схожі з .NET класи на інші платформи, включаючи власний компілятор C# та CLR (Common Runtime Language). Сьогодні Mono підтримує майже всі функції .NET і все ще збільшує свої можливості [77].

Виконання коду для цього фреймворку також аналогічне .NET Framework:

- Програміст пише код на C #.

- C# код компілюється за допомогою вбудованого компілятора C# і генерує байт-код або проміжну мову. Це є збірка (build).

- Збірка працює на цільових пристроях, використовуючи пакетне середовище моно-режиму виконання або загальну мову виконання (Common Language Runtime). Mono виконує важливі роботи - це збирання сміття, безпека, обробка винятків тощо.

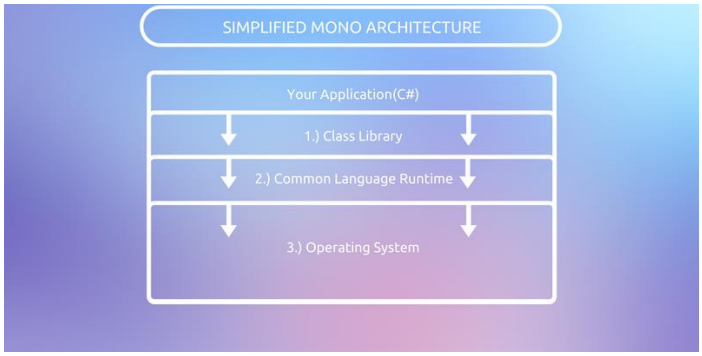


Рисунок 2.21. Архітектура Mono [77].

Mono's Code Execution Engine: mono runtime містить механізм виконання коду, який перетворює байт-код (IL) у нативний код. Двигун виконання коду піддається трьом режимам наступним чином:

1) Just in Time (JIT) компіляція: під час виконання коду програма перетворить байт-код у нативний код.

2) Ahead of Time (AOT) компіляція: цей режим роботи збирає більшу частину коду до запуску коду.

3) Повна статична компіляція (Full Static): небагато пристроїв підтримують такий режим компіляції, серед них IOS, PlayStation 3 та XBOX. Це робить компіляцію AOT на крок попереду.

Крос-платформна функція Mono - одне з головних рішень для постачальників програмного забезпечення, орієнтованих на підтримку широкого кола популярних платформ, доступних на ринку. Наприклад, це реалізована в ігровому рушії Unity.

Як Unity підтримує функцію крос-платформи

Unity підтримує можливість крос-платформи, це означає зробити збірку один раз та розгортати її де завгодно, це робить ігровий рушій Unity більш переважним серед інших. Для цього в комплекті з Unity йде постачання Mono.

Для того, щоб написана програма, яка була створена на Unity (за допомогою Mono), могла запрацювати, наприклад, на пристрої с операційною системою Android, «Mono Runtime Environment» поєднується зі збіркою, яка відповідає за подальші дії віртуальної машини для самостійного запуску коду

на іншій платформі. Mono конвертує складений байт-код у збірці у нативний код. [78]

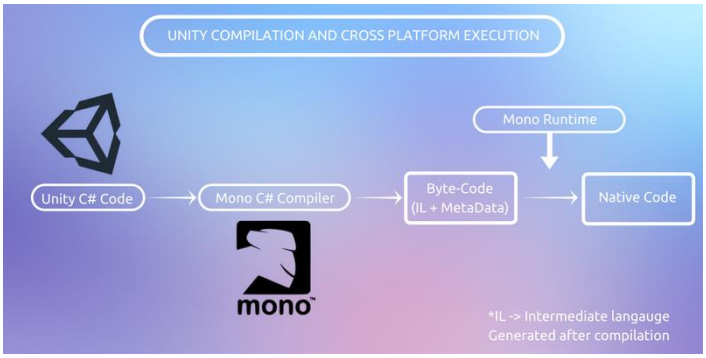


Рисунок 2.22. Компіляція Unity для крос-платформи [78].

На рисунку 2.22 можна побачити компіляцію Unity для крос-платформи. Потрібно сказати, що Mono не додає весь фреймворк до збірки, невикористані класи знімаються, і додається єдина частина фреймворку, яка використовується написаним додатком. Тому Unity створює збірки з більшими розмірами, і це виправдано, оскільки це потрібно для підтримки крос-платформної функції.

Mono приносить крос-платформні .NET-сумісні функції для Unity, програмісти пишуть код на мові програмування C#, а коли роблять збірку, він додає Mono Runtime. Подальше виконання режиму Mono запускає розроблений додаток, виступаючи як віртуальна машина на будь-якій платформі [78].

2.7 Підтримка профілю .NET в Unity

Unity підтримує ряд профілів .NET. Кожен профіль забезпечує іншу поверхню API для коду C#, який взаємодіє з бібліотеками класів .NET.

Спадковий сценарій виконання

Спадковий сценарій виконання підтримує два різних профілі: .NET 2.0 Subset та .NET 2.0. Обидва вони тісно узгоджуються з профілем .NET 2.0 від Microsoft. Профіль підмножини .NET 2.0 менший, ніж профіль .NET 4.x, і він дозволяє отримати доступ до API бібліотеки класів, якими користується більшість проектів Unity. Це ідеальний вибір для обмежених розмірами платформ, таких як мобільні пристрої, і він пропонує набір портативних API

для підтримки багатьох платформ. За замовчуванням більшість проектів Unity повинні використовувати профіль .NET Standard 2.0. [79]

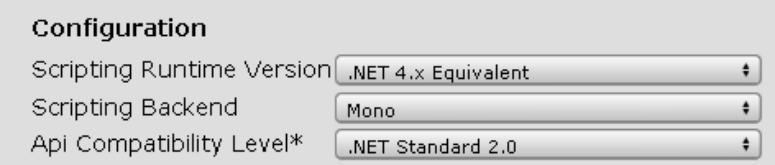


Рисунок 2.23. Настройки конфігурації у Unity.

Як можна побачити на рисунку 2.23, для написання системи я використовував, у програмі Unity, наступні настройки конфігурації: Scripting Runtime Version: .NET 4.x Equivalent та Api Compatibility Level: .NET Standard 2.0. А для компіляції: Mono.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі були розглянуті технології для розробки системи віртуальної реальності. Була розглянута стаціонарна ВР та мобільна ВР. Були розглянуті пристрої управління ВР. Детально була розглянута платформа Unity для створення системи ВР, з чого вона складається, які вікна для роботи розробників вона містить, які можливості вона дає для розробників. Були розглянуті функціональні можливості та технічні характеристики Unity. Було розглянуто, які особливості та переваги має Unity для створення проектів ВР та чому для створення систем ВР краще за все обрати ігровий рушій Unity.

Був розглянутий пакет Google VR SDK, який допомагає розробникам при створенні проектів ВР, який дає можливість симулювати процес ВР у самому редакторі Unity. Були розглянуті об’єкти, які цей пакет в собі містить і було розглянуто для чого вони потрібні під час створення системи ВР. Було розглянуто, як створювати проекти ВР у Unity, що саме для цього потрібно, які компоненти для об’єктів використовувати і як взагалі працювати с цими компонентами. Були розглянуті найпопулярніші проекти ВР створенні на платформі Unity.

Була розглянута мова програмування C#, яка використовується у Unity для написання логіки системи. Була розглянута платформа .NET та її особливість. Була розглянута платформа .NET Framework, її основні компоненти. Були розглянуті етапи компіляції та виконання програм, написаних за допомогою .NET Framework.

Була розглянута платформа Mono та її архітектура. Була детально розглянута крос-платформна функція Mono та як Unity підтримує, завдяки неї, функцію крос-платформи. Була розглянута компіляція Unity для крос-платформи. Була розглянута підтримка профілю .NET в Unity та спадковий сценарій виконання.

РОЗДІЛ 3
ДЕТАЛІ РОЗРОБКИ СИСТЕМИ

3.1 UI у системі ВР

Користувацькі інтерфейси у системі ВР є достатньо важливою темою. Оскільки, користувачу даної системи повинно бути зручно з ними взаємодіяти. Користувачу взаємодія з UI повинна бути комфортною і легкозрозумілою.

У Unity є можливість для адаптованої верстки таких інтерфейсів, використовуючи компонент Grid Layout Group (Рисунок 3.1) для батьківського об’єкту, наприклад, для об’єкту Canvas та компонент Layout Element (Рисунок 3.2) для його дочірніх об’єктів. Ці компоненти використовуються переважно для мобільних VR додатків, вони дозволяють зробити елементи Canvas адаптовані, тобто на різних смартфонах їхнє розташування буде виглядати однаковим [80].

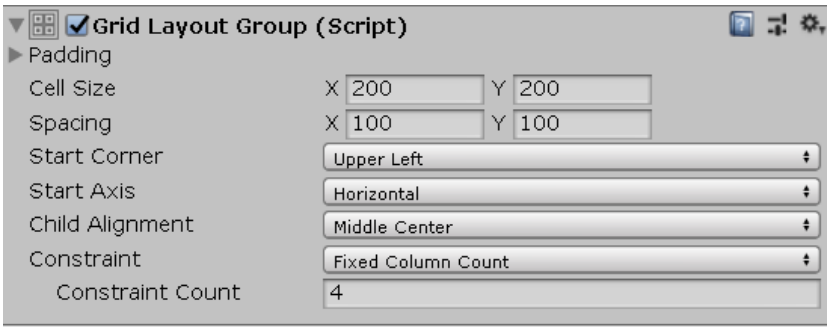


Рисунок 3.1. Компонент “Grid Layout Group” на батьківському об’єкті.

Рисунок 3.2. Компонент “Layout Element” на дочірньому об’єкті.

У звичному мобільному додатку, щоб взаємодіяти з елементами UI-інтерфейсу необхідно використовувати пальці і екран смартфону. У

мобільному VR взаємодіяти з UI можливо шляхом наведення погляду на UI-елемент, завдяки точки, яка виступає як курсор перед очима користувача. Тобто, якщо в звичному мобільному додатку, щоб натиснути кнопку, користувачі пальцями натискають на цю кнопку на екрані телефона, то у мобільному VR додатку для того, щоб натиснути кнопку достатньо лише навести на неї погляд. Реалізувати взаємодію між користувачем та UI-елементами можна декількома шляхами.

Перший, це просто навести погляд на інтерактивний UI-елемент і запланована дія відразу відбудеться. Такий спосіб є достатньо швидким, але користувач може випадково навести погляд і відбудеться запланована дія, хоча він цього міг і не хотіти зробити.

Тому для вирішення цієї проблеми існує другий шлях, це наведення погляду разом із заповненням Progress Bar. Тобто, при наведенні погляду на інтерактивний UI елемент дія не відразу відбудеться, а спочатку необхідно затримати свій погляд на цьому елементі і дочекатися заповнення прогресс бару і тільки після цього відбудеться запланована дія.

Окрім цього, якщо користувач випадково наведе погляд на інтерактивний елемент, то в нього є час відвести погляд від нього поки не заповниться прогрес бар (зазвичай це одна секунда). Також, деякі мобільні платформи мають спеціальну кнопку на своїх шоломах, тобто можна навести погляд на інтерактивний елемент і дія відбудеться лише тоді, коли користувач натисне цю кнопку тримаючи погляд на елементі. Нова версія Samsung Gear VR має спеціальний контролер, на якому знаходяться кнопки і користувач може використовувати цей контролер для взаємодії з елементами UI.

Комп’ютерні VR платформи мають спеціальні контролери, джойстики та інші гарнітури для взаємодії з UI. Що є достатньо швидким і простим засобом для взаємодії користувача з UI у віртуальному додатку.

Так як у мене система віртуальної реальності практично для будь-яких мобільних пристроїв операційної системи Android, то для зручної та простої

взаємодії між користувачем і UI я використовую рішення з наведенням погляду користувача у другому варіанті переміщення та відведення погляду. А також з наведенням погляду та з заповненням Progress Bar.

3.2 Сцени та головні об’єкти системи.

Система складається з декількох сцен та багатьох об’єктів. Перша сцена створена для першого варіанту переміщення, друга сцена створена для другого варіанту переміщення, третя сцена для гри на тренування пам’яті. Також, була створена сцена головного меню.

Головними об’єктами першої сцени є:

- 3D модель самого лабіринту;
- Об’єкт камера, яка виступає головою користувача;
- UI для відображення точок переміщення;
- UI для відеозапису дій користувача;
- Об’єкти телепортів, які виступають у якості точок переміщення;
- Об’єкти для повернення на початок лабіринту;
- Об’єкт кінця лабіринту.

Головними об’єктами другої сцени є:

- 3D модель самого лабіринту;
- Об’єкт камера, яка виступає головою користувача;
- 3D об’єкт капсула (capsule), яка виступає у якості тіла користувача;
- UI для відображення точок переміщення;
- UI для відеозапису дій користувача;
- UI для другого варіанту переміщення;
- Об’єкти точки переміщення;
- Об’єкти для повернення на початок лабіринту;
- Об’єкт кінця лабіринту.

Головними об’єктами третьої сцени є:

- Об’єкт камера, яка виступає головою користувача;

- UI в якому розташовані картки;
- UI для перезавантаження гри;
- UI для виходу у головне меню системи.

Головними об’єктами сцени головного меню є:

- Об’єкт камера, яка виступає головою користувача;
- UI для переключання між сценами;
- UI для вибору варіанту гри на пам'ять;
- UI для відображення карти лабіринту з точками переміщення.

3.3 Варіанти переміщення користувача та їх розробка

Оскільки, ця система розроблена для мобільної VR, щоб вона була доступною для якомога більшої кількості користувачів, а у ВР для мобільних пристроїв враховуються лише нахили голови, а не рух людини у реальному просторі, то в цій системі я реалізував два варіанти переміщення користувача. Користувач у головному меню може обрати варіант переміщення, який йому більш зручніший.

Перший варіант переміщення

Перший варіант переміщення користувача по віртуальному лабіринті полягає у наведені погляду на спеціальні телепорти, які розташовані на підлозі віртуального лабіринту і виступають у якості точок для переміщення.

Користувачеві необхідно навести погляд на телепорт, в місце якого він планує переміститися, спеціальною точкою, яка виступає у якості курсору і яка знаходиться перед очима користувача у віртуальних окулярах. Після наведення погляду користувача на телепорт йде заповнення Progress Bar (буквально одну секунду) і після заповнення відбувається переміщення користувача у потрібну точку. Під час переміщення відбувається операція інкремент вибраної точки, для того, щоб мати можливість перевірити скільки разів користувач під час проходження лабіринту знаходився у цій точці.

Progress Bar - є дуже важливою частиною для зручної взаємодії користувача між інтерактивними об’єктами. Progress Bar це об’єкт, який починає заповнюватись, коли користувач навів свій погляд на інтерактивний об’єкт. Progress Bar використовується для того, щоб якщо користувач випадково навів погляд на інтерактивний об’єкт міг відмінити заплановану дію.

Для реалізації першого варіанту я використовую клас ImageProgressBar, для реалізації заповнення Progress Bar. Також використовую клас CalculateResultPoints для роботи з текстом UI з підрахунком точок переміщення, який змінює відображення нового тексту підрахунку точок, коли користувач рухається по лабіринту через телепорти. Також, використовую клас TeleportToPosition, який містить метод MovePlayer(). Щоб метод MovePlayer() спрацював, я наклав на кожну із точок скрипт TeleportToPosition та викликаю його метод MovePlayer() тоді, коли користувач навів погляд на інтеркативний об’єкт коло та дочекався заповнення Progress Bar. В свою чергу метод MovePlayer() викликає статичний метод SetPosition() класу Player, в який передається позиція точки, на місце якої користувач планує переміститися, а також відбувається збільшення значення точки у UI з підрахунком точок, тобто відбувається операція інкремент.

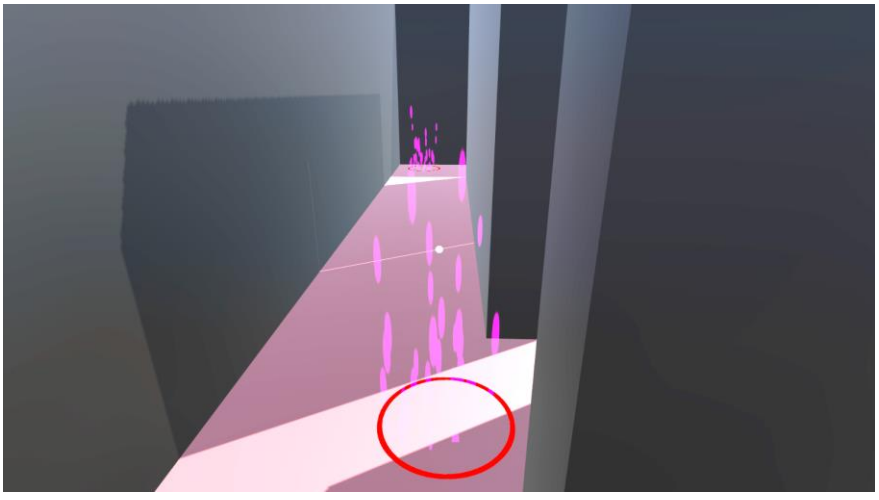


Рисунок 3.3. Перший варіант переміщення.

На рисунку 3.3 ми можемо бачити сцену першого варіанту переміщення через віртуальний лабіринт. Також, ми можемо побачити спеціальні телепорти,

на які потрібно наводити погляд користувачеві, щоб він міг переміщатися по віртуальному лабіринту.

3.4 Розробка точок для переміщення

У якості точок для переміщення я використовую спеціальні телепорти, які розташовані по віртуальному лабіринту. Вони складаються з об’єктів:

1. Circle
2. Arrow
3. Particle System

Об’єкт Circle – це інтерактивний об’єкт, картинка, елемент UI у редакторі Unity. У телепорті ця картинка виступає у вигляді червоного кругу.

Об’єкт Arrow – це також картинка у вигляді стрілки і вона виступає у якості Progress Bar, який заповнюється під час наведення погляду користувача на об’єкт Circle.

Об’єкт Particle System – це об’єкт, який містить компонент “Particle System”.

У системі я використовую компонент Particle System для створення ефекту справжнього телепорту, для того, щоб ці телепорти здавались користувачу більш справжніми. Компонент Particle System має велику кількість параметрів для створення абсолютно різних ефектів у редакторі Unity [81].

Для телепортів я додав матеріал Particles, які є овальними об’єктами і схожі на вогники, я вибрав форму конуса, налаштував необхідний радіус, зробив появу ефектів випадковою, змінив їх кількість та час їх появи, змінив їх швидкість у часі, змінив їх час життя, колір та багато іншого. Всі ці зміни дали ефект постійної появи великої кількості вогників з кола, що створюють ефект телепорту.

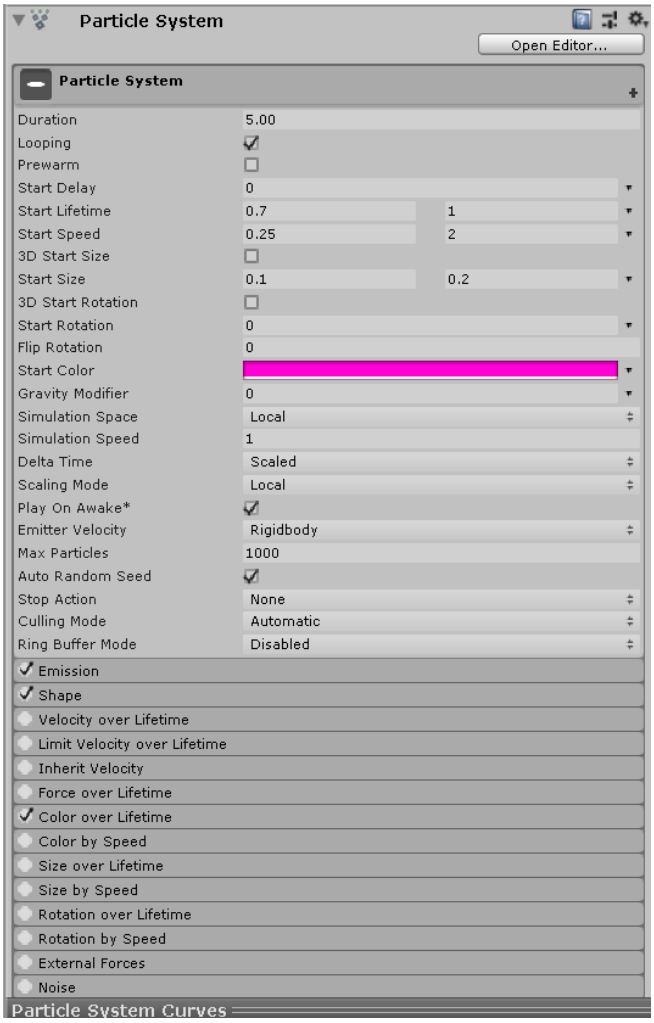


Рисунок 3.4. Particle System

На рисунку 3.4 можемо побачити частину моїх налаштувань компоненту Particle System для об’єктів телепортів.

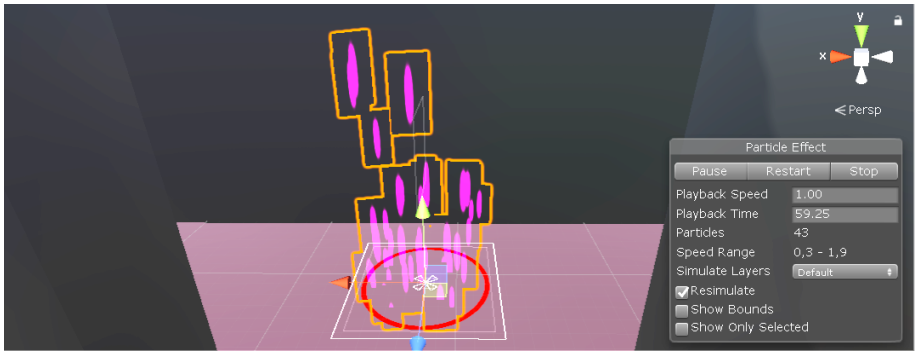


Рисунок 3.5. Демонстрація телепортів у сцені Unity.

На рис. 3.5 можемо бачити саму демонстрацію телепортів у сцені Unity, а праворуч знизу вікно з настройками “Particle Effect” для роботи зі створеними ефектами.

Другий варіант переміщення

Другий варіант переміщення був реалізований таким чином, що користувачу потрібно наводити погляд на одну із чотирьох стрілок і користувач почне пересуватися у напрямку цієї стрілки по віртуальному лабіринту.

В даному випадку процес пересування користувача є більш плавним та реалістичним, оскільки не відбувається різкого переміщення (телепортації) як у першому варіанті пересування.

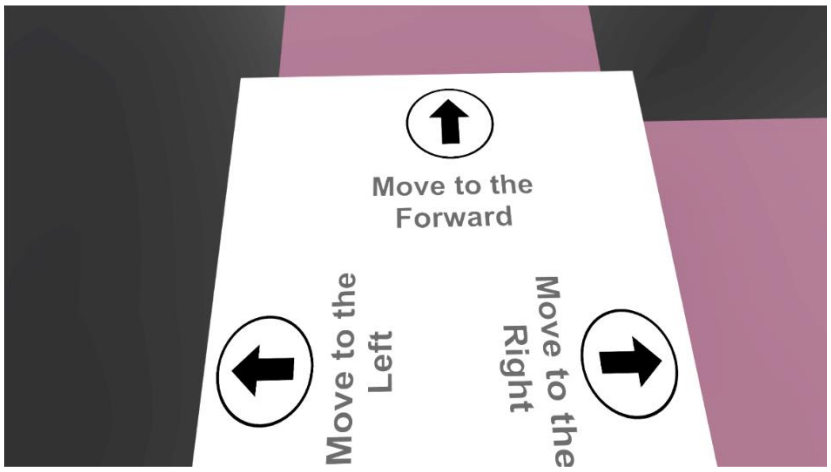


Рисунок 3.6. Другий варіант переміщення (вигляд спереду).

На рисунок 3.6 показаний вигляд спереду другого варіанту руху через віртуальний лабіринт. Під стрілкою є напис, щоб користувач міг легко зрозуміти, в якому напрямку він почне пересуватися після наведення погляду на стрілку.

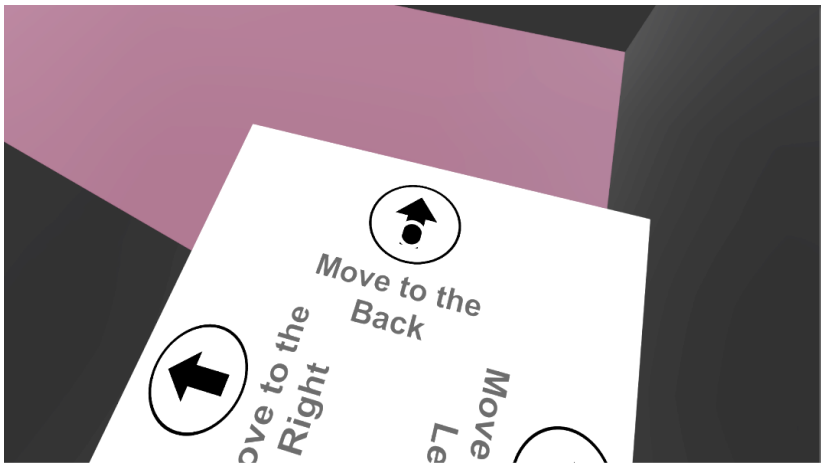


Рисунок 3.7. Другий варіант переміщення (вид ззаду).

На рисунку 3.7 показаний вид ззаду другого варіанту переміщення через віртуальний лабіринт.

Щоб побачити стрілки, які розташовані ззаду користувача, користувачеві потрібно озирнутися головою назад і опустити погляд вниз, а для самого процесу руху користувачеві потрібно подивитися на стрілку.

Під час пересування користувача по віртуальному лабіринту другим варіантом, також відбувається підрахунок точок, на яких був присутнім користувач. Для цього я використовував об’єкти з компонентом Capsule Collider, які виступають як точки в яких знаходився користувач під час пересування та розставив їх по лабіринту як у сцені першого варіанту переміщення. Цей компонент необхідний у Unity для того, щоб зрозуміти, що користувач був на ньому присутнім. Тілом користувача виступає 3D об’єкт капсула, яка містить компонент Capsule Collider. Також, на цей об’єкт я додав компонент Rigidbody [82], що дозволяє зробити об’єкт твердим тілом і це дає можливість, щоб користувач не міг проходити крізь стіни під час пересування.

Коли користувач під час пересування по віртуальному лабіринту входить у межі Collider об’єкту точки, то відбувається метод OnTriggerEnter(Collider collider) класу PointsPlaces, який збільшує значення точки переміщення і це можна побачити на UI з підрахунком точок переміщення. Для того, щоб метод OnTriggerEnter(Collider collider) виповнився, у об’єктах точок в компоненті Capsule Collider необхідно виставити галочку біля значення Is Trigger. Також, для кожного з таких об’єктів я наклав скрипт з назвою PointsPlaces. UI з підрахунком точок переміщення також оновлюється у режимі реального часу як і у першому варіанті переміщення.

Звичайно, всі ці об’єкти є невидимі для користувача, щоб не псувати йому повне занурення у віртуальний простір.

Всі значення точок переміщення зберігаються у сховищі. Для цього я використовую готовий клас PlayerPrefs у Unity, який виступає як сховище [83]. За допомогою цього класу можна зберігати дані на самому пристрої

користувача, тобто на мобільному телефоні з ОС Android. Для збереження даних у сховищі використовується статичний метод SetString(string key, string value) класу PlayerPrefs, в якому в перший аргумент задається ключ, а у другий значення, яке необхідно зберігти у сховищу. Для того, щоб взяти значення з сховища використовується статиний метод GetString(string key), в якому в аргумент потрібно вказати створений ключ.

3.5 Розробка UI для пересування

Для другого варіанту переміщення я створив UI зі стрілками у редакторі Unity. Для пересування вперед користувачу потрібно навести погляд на стрілку з написом “Move to the Forward”, для пересування вправо користувачу потрібно навести погляд на стрілку з написом “Move to the Right”, для пересування вліво користувачу потрібно навести погляд на стрілку з написом “Move to the Left”, для пересування назад користувачу потрібно навести погляд на стрілку з написом “Move to the Back”.

Для такої реалізації на кожну стрілку я додав компонент Event Trigger та її типи Pointer Enter та Pointer Exit і для кожного типу додав метод скрипта класу написаного мною. Це клас PlayerMove власне і сам скрипт має таку назву, бо у Unity назва скрипта повинна співпадати із назвою класу.

Клас PlayerMove для руху користувача вперед містить метод MoveToForward(), для руху вправо мотод MoveToRight(), для руху назад метод MoveToBack(), для руху вліво метод MoveToLeft().

Метод у Pointer Enter відбудеться відразу як користувач наведе погляд на стрілку, а метод у Pointer Exit відбудеться відразу як користувач відведе погляд від стрілки.

Також, у класі PlayerMove я використовую зміну trigger, значення якої змінюється коли користувач наводить погляд на одну із стрілок і це дає зрозуміти на яку саме стрілку користувач навів погляд і яку операцію для цього необхідно виконувати.

Опис змінних та методів першого варіанту переміщення

Тут наведені деякі зміні та методи першого варіанту переміщення по віртуальному лабіринту.

1. *timeToFill* – це час для заповнення Progress Bar.

2. *startTime* – це початковий час.

3. *overTime* - це час за який повинен заповнитись Progress Bar від початку часу.

4. *progressBarImage.fillAmount* - значення картинки для її заповнення в межах [0;1]. [84]

5. *Player.transform.position* - координати позиції користувача [85].

6. *new Vector3(interectiveObject.position.x, 1.8, interectiveObject.position.z)* - створення нового об’єкту для зміни координат користувача, де:

- *interectiveObject.position.x* - це значення координати X інтерактивного об’єкту,
- *1.8* - це значення координати Y інтерактивного об’єкту,
- *interectiveObject.position.z* - це значення координати Z інтерактивного об’єкту.

Для коректного заповнення Progress Bar використовується наступна формула:

$$fillAmount = Math.Lepr(0,1,\frac{Time.time - startTime}{timeToFill})$$

Math.Lepr(float a, float b, float t) - Лінійно інтерполює між a і b на t. Параметр t обмежений діапазоном [0, 1]. [86]

Також, під час виконання цього сценарію (клас ImageProgressBar), у програмному коді використовується методи StartCoroutine() та StopCoroutine().

Метод StartCoroutine() – це метод, який дозволяє виконувати функцію паралельно протягом деякого часу, дозволяє призупинити виконання та автоматично відновити на наступному кадрі у Unity [87].

Метод StopCoroutine() – це метод, який використовується для призупинення паралельного виконання методу StartCoroutine().

Опис змінних та методів другого варіанту переміщення

Тут наведені деякі зміні та методи другого варіанту переміщення по віртуальному лабіринту.

1. Зміна *trigger* - параметр для визначення в якому напрямку рухатись користувачеві.

2. Player.transform.Translate (Vector3 translation * Time.deltaTime) - це метод для переміщення користувача у потрібному напрямку де: [88]

- *Vector3 translation* - напрямок руху по певній осі.
- *Time.deltaTime* – це час завершення в секундах з моменту останнього кадру. [89] Тобто, переміщення відбувається у потрібному напрямку на одну одиницю за секунду.

3.6 Розробка UI для повернення на початок лабіринту

Якщо користувач потрапить у глухий кут, то він може навести погляд на спеціальний інтерфейс і після заповнення Progress Bar він повернеться на початок лабіринту.

Для розробки такого UI я використовував компонент об’єкт Canvas та два об’єкти Image, які є картинками та дочірніми об’єктами об’єкту Canvas.

Canvas – об’єкт у редакторі Unity для відображення UI елементів, таких як текст, картинки, кнопки та інше [90]. Одна з картинок виступає як інтерактивний об’єкт, на яку користувачеві необхідно навести погляд для початку заповнення Progress Bar. Ця картинка має напис о поверненні на початок лабіринту, а інша картинка виступає як Progress Bar, тобто вона почне заповнюватись після наведення погляду. Ця можливість була реалізована для швидкого повернення на початок пересування і це може бути зручніше для користувача.

На рис. 3.8 ми можемо побачити UI та процес заповнення Progress Bar, це картинка білого кольору, після заповнення якої користувач повернеться на початок лабіринту.

Рисунок 3.8. UI для повернення на початок лабіринту.

Для реалізації я використовую клас ImageProgressBar для картинки, яка виступає як Progress Bar, а також метод GoToTheStart() класу Restart для повернення на початок лабіринту. Також, в свою чергу метод GoToTheStart() викликає статичний метод SetPosition() класу Player для зміни позиції користувача.

3.7 Розробка UI для відеозапису

Точки які оновлюються під час переміщення користувача дають лише інформацію де знаходився користувач під час проходження лабіринту. Але для того, щоб мати можливість повністю передивитися як людина вела себе під час проходження лабіринту у цій системі є можливість повного відеозапису його проходження. Тобто, перед проходженням можна запустити відеозапис, а після завершення проходження лабіринту зберегти цей запис. Цей запис збережеться у галереї користувача на його мобільному пристрої. І вже після цього його можна переглянути.

Для цього я створив UI під назвою Video Recorder (рисунок 3.9), який містить три інтерактивні об’єкти, Start Record, Save Video та Go to Menu. Після

наведення погляду користувача та заповнення Progress Bar на одному з об’єктів відбудеться запрограмована дія.

Для об’єкту Start Record, почнеться відеозапис проходження користувача. За це виконання відповідає метод StartVideo() у класі GameplayRecorder. В свою чергу метод StartVideo() викликає метод StartRecord() класу RecordManager.

Для об’єкту Save Video, відбудеться збереження відеозапису у галерею користувача. За це виконання відповідає метод SaveVideo() класу GameplayRecorder, який в свою чергу викликає метод StopRecord() класу RecordManager.

Для об’єкту Go to Menu, відбудеться повернення до головного меню користувача. За це виконання відповідає метод GoToMenu() класу ChangeScenes.

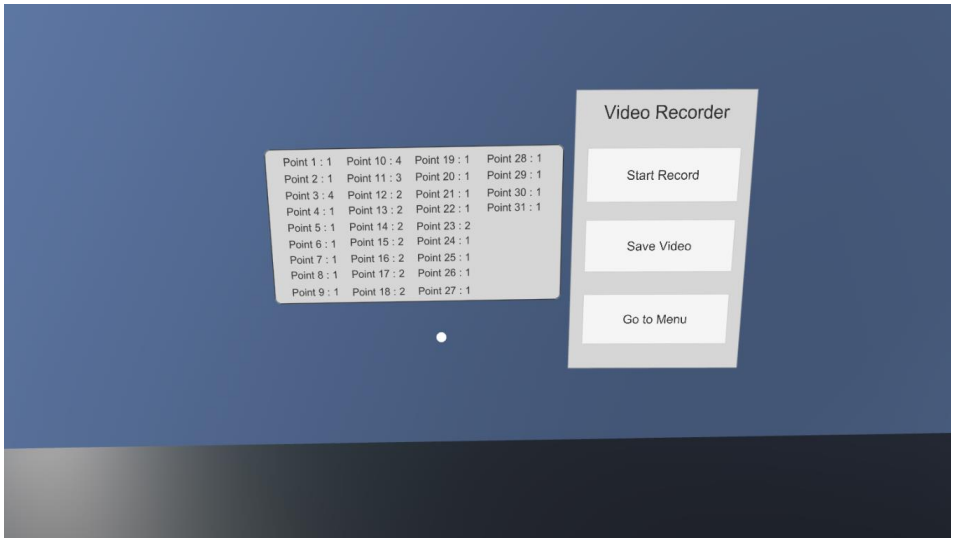


Рисунок 3.9. UI з обчисленням точок та UI Video Recorder.

На рисунку 3.9 зліва видно UI обчислених точок переміщення, яке оновлюється в режимі реального часу. Праворуч - UI відеозапису, за допомогою якого користувач може виконувати відеозапис та записувати всі власні дії під час пересування по лабіринту.

У першому варіанті переміщення, після того як користувач переміститься на місце телепорту, разом із цим відбувається операція інкремент відповідної змінної. Тобто, наприклад, якщо користувач двічі за проходження віртуального

лабіринту опиниться у глухому куті під номером 9, то на UI з обчисленням точок значення під точкою номер 9 буде рівне двом.

3.8 Розробка гри для тренування пам’яті користувача

Оскільки, одним з деяких шляхів для запобігання прогресії хвороби Альцгеймера є тренування пам’яті хворого [1], то в цій системі я також реалізував можливість для тренування пам’яті користувача.

Для цього була розроблена гра для тренування пам’яті, яка складається з декількох пар карток, які розташовані в перемішку і користувач повинен під час гри запам’ятовувати де знаходиться та чи інша картка, відкриваючи одну з карток знайти серед інших карток для неї пару. Якщо друга картка не збіглася, то вони обидві закриваються і гра продовжується, якщо збіглася, то ці картки зникають. Гра продовжується поки користувач не знайде усі пари карток.

Для реалізації цієї гри я створив UI з картками, для цього я використовував об’єкт Canvas, дочірній до нього об’єкт Panel та об’єкти Image. Для об’єкту Panel додав компонент Grid Layout Group, щоб дочірні до нього об’єкти Image були адаптовано розташовані для різних розширень екрану смартфону.

Логіка гри була написана у класі BtnLogic та GameController. Для пошуку карток в сцені був написаний метод GetButtons() класу GameController. Метод AddGamePuzzles() добавляє необхідну кількість карток у список. Метод Shuffle() випадково виставляє картки у сцені, щоб кожного разу картка відображала різні картинки, а не постійно одну і ту саму. Ці методи відбуваються в самому початку гри.

У класі BtnLogic() був написаний метод PickAPuzzle(), який відбувається, коли користувач навів погляд на одну із карток і дочекався заповнення Progress Bar. Метод PickAPuzzle() дає можливість відкрити не більше двох картинок карток у сцені, після чого відбувається метод CheckIfThePuzzleMatch() класу GameController, який перевіряє чи є картки однаковими. Якщо вони не

співпадають, то картки закриються і гра продовжиться, якщо співпадають, то вони пропадуть і буде викликаний метод CheckIfTheGameIsFinished() класу GameController, який перевіряє чи всі пари карток були знайдені. Якщо так, то відкриється UI з результатом гри користувача. Користувач може бачити за скільки ходів він зміг знайти всі пари карток, тобто користувач отримав добрий результат чи ні.

Також, у цій системі було реалізовано 3 варіанти цієї гри:

1) на 4 картки - це самий простий варіант гри;

2) на 8 карток - це більш складний варіант гри, бо користувачеві відповідно потрібно буде запам’ятовувати більше карток;

3) та остання гра на 12 карток - є самим складним варіантом гри.

Якщо користувач, починаючи з простого рівня, буде кожного дня грати у цю гру і покращувати свою пам'ять, то зможе переходити до більш складних рівнів. Ця гра є популярною та дуже ефективною для тренування пам’яті.

Рисунок 3.10. Гра для тренування пам'яті на 12 карток.

На рисунку 3.10 ми бачимо ігровий процес гри для тренування пам'яті на 12 карток.

Як бачимо, дві карти відкриті, але вони не однакові, тому вони будуть закриті знову і гра продовжиться. Як ми бачимо, гра дуже яскрава, тому вона корисна не тільки тому, що вона тренує пам’ять користувача, але і досить яскрава, а отже, це також буде піднімати настрій користувачу.

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі були наведені деталі розробки системи. Була розглянута важливість створення комфортних UI у системі ВР та як у Unity можна створювати адаптовані UI для різної роздільної здатності екрану смартфону.

Були розглянуті сцени та головні об’єкти системи. Були детально розглянуті варіанти переміщення користувача по віртуальному лабіринту та їх розробка. Була описана важливість Progress Bar.

Були розглянуті класи та їх методи першого варіанту переміщення. Був розглянутий UI з підрахунком точок пересування та було розглянуто, як відбувається підрахунок точок під час переміщення користувача в обох варіантах по віртуальному лабіринту. Була розглянута розробка точок для переміщення (телепортів), з чого вони складаються. Був розглянутий компонент Particle System, який дає можливість для створення ефектів та його використання для телепортів.

Був розглянутий другий варіант переміщення, розробка UI для переміщення у другому варіанті. Були описані класи та їх методи, які використовуються для реалізації другого варіанту переміщення. Були розглянуті алгоритми варіантів переміщення, були наведені їх рисунки та було описано їх зміні та методи.

Була розглянута розробка UI для повернення на початок лабіринту, з яких об’єктів складається та які методи класів використовує. Була розглянута розробка UI для відеозапису, з яких об’єктів складається та методи яких класів використовує.

Була розглянута гра для тренування пам’яті користувача. Було розглянуто з яких об’єктів вона складається. Були розглянуті написані класи та їх методи для логіки гри. Була описана логіка цієї гри та були розглянуті варіанти цієї гри.

Були продемонстровані скріншоти розробки системи для наочної демонстрації.

РОЗДІЛ 4
АНАЛІЗ РОЗРОБЛЕННОЇ СИТЕМИ

4.1 Як взаємодіяти із системою

Для того, щоб користувач міг скористатися системою віртуальної реальності для визначення дій людини у лабіринті, йому необхідно завантажити apk файл розробленої системи на свій смартфон з операційною системою Android з версією не меншою ніж 5.0. Також, для занурення у віртуальний простір користувачу необхідно мати окуляри віртуальної реальності для мобільних пристроїв, це може бути звичайний Cardboard. Після завантаження додатку, необхідно помістити смартфон в окуляри та можна розпочати використовувати систему.

Аналіз системи був проведений на смартфоні Samsung Galaxy S9+ з ОС Android 10.1.

4.2 Взаємодія з головним меню

Після запуску додатку користувач побачить UI головного меню додатку (Рисунок 4.1), в якому, завдяки наведенню погляду, може обрати варіант переміщення по віртуальному лабіринту, гру для тренування пам’яті, подивитися карту лабіринту з точками переміщення та побачити результати.

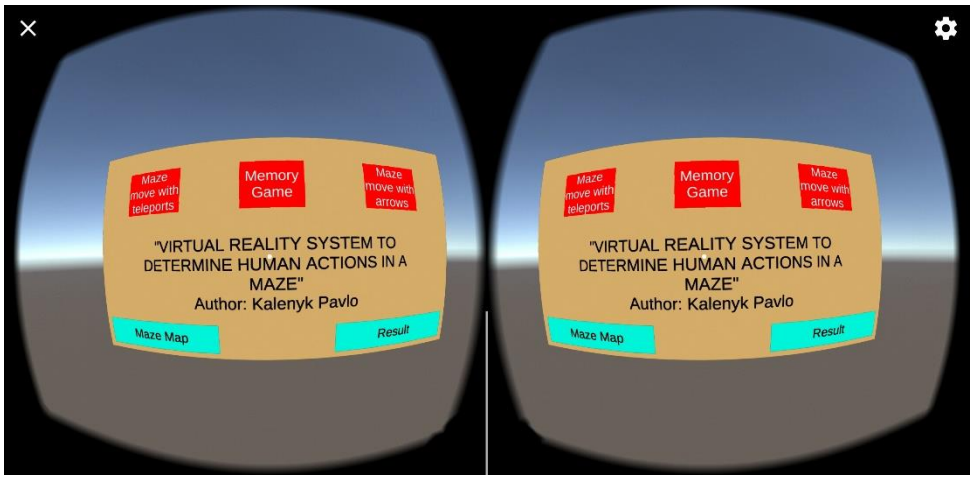


Рисунок 4.1. UI головного меню.

На рис. 4.1 можемо побачити головне меню додатку, яке виступає у якості UI. Цей інтерфейс містить п’ять елементів, які являють собою інтерактивні

об’єкти, це: Maze move with teleports, Memory Game, Maze move with arrows, Maze Map та Result.

Після наведення погляду на об’єкт Maze move with teleports відбудеться заповнення Progress Bar після чого користувачу відкриється сцена віртуального лабіринту з першим варіантом переміщення.

Після наведення погляду на об’єкт Maze move with arrows відбудеться заповнення Progress Bar після чого користувачу відкриється сцена віртуального лабіринту з другим варіантом переміщення.

Після наведення погляду на об’єкт Memory Game відбудеться заповнення Progress Bar після чого користувачу відкриється новий UI з вибором варіанту гри для тренування пам’яті. Є три варіанти гри для тренування пам’яті, на 4 картки, на 8 карток та на 12 карток.

Після наведення погляду на об’єкт Maze Map відбудеться заповнення Progress Bar після чого користувачу відкриється новий UI з рисунком карти лабіринту та точок переміщення, які розставлені по лабіринту.

Після наведення погляду на об’єкт Result відбудеться заповнення Progress Bar після чого користувачу відкриється новий UI з результатами проходження.

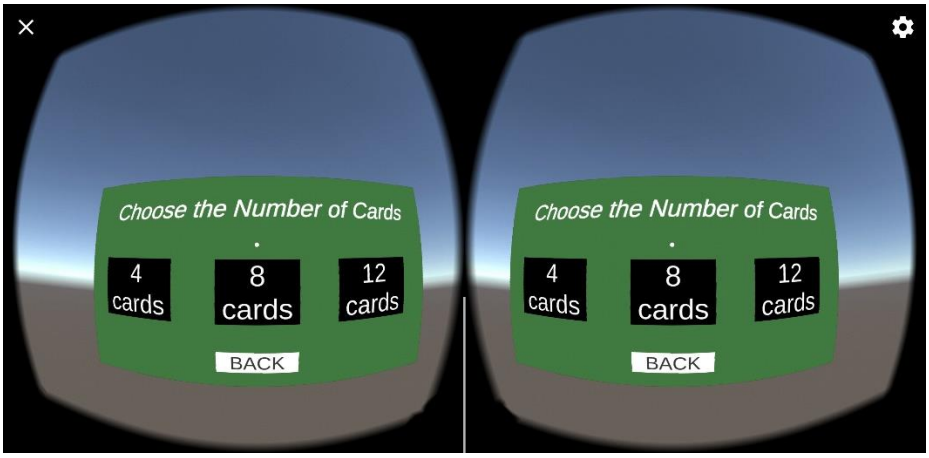


Рисунок 4.2. UI з вибором режиму гри для тренування пам’яті.

На рис. 4.2 можна бачити варіанти гри для тренування пам’яті. Після наведення погляду користувача на них відбудеться заповнення Progress bar після чого відкриється сцена з вибраним варіантом гри.

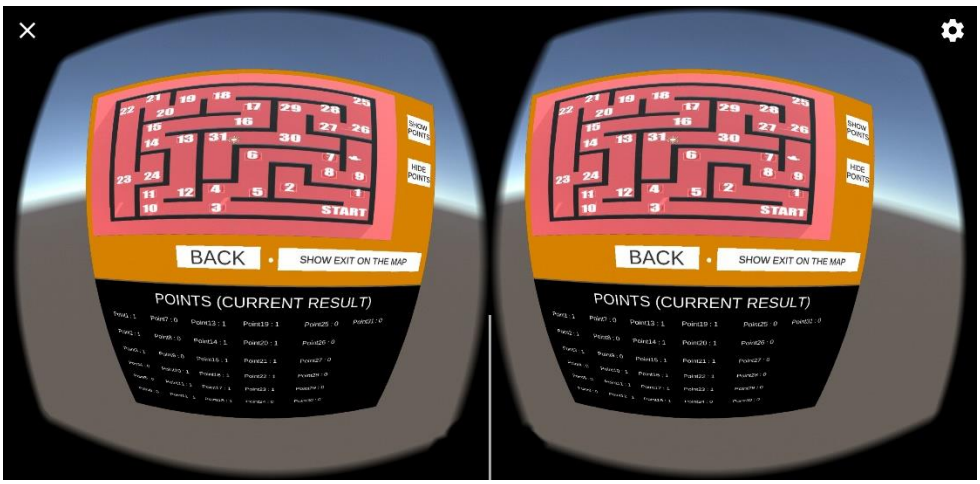


Рисунок 4.3. UI з картою лабіринту та з точками переміщення.

На рис. 4.3 можна побачити UI, який відобразиться користувачу, коли він наведе погляд у головному меню на інтерактивний об’єкт Maze Map.

Цей UI містить карту лабіринту з пронумерованими точками для переміщення, на цій карті можна побачити де та яка точка розміщена та назву START. START вказує на початок віртуального лабіринту, з якого користувач починає свій рух.

Також, цей UI містить інтерактивний об’єкт BACK, для повернення у головне меню, інтерактивний об’єкт SHOW EXIT ON THE MAP, який відобразить вихід із лабіринту, якщо користувач наведе на нього свій погляд та дочекається заповнення Progress Bar. Вихід спочатку скривається для того, щоб користувач не міг побачити де знаходиться сам вихід, якщо, наприклад, випадково відкриє цей UI.

Цей UI також містить інтерактивний об’єкт SHOW POINT, після наведення погляду користувача на нього на заповнення Progress Bar, відобразиться UI з точками пересування останнього проходження лабіринту користувача. UI з точками останнього пересування можна побачити знизу на рис. 4.3. UI також містить об’єкт HIDE POINTS. Якщо користувач наведе погляд на інтерактивний об’єкт HIDE POINTS та дочекається заповнення , то UI з точками пересування останнього проходження лабіринту користувача буде закрито.

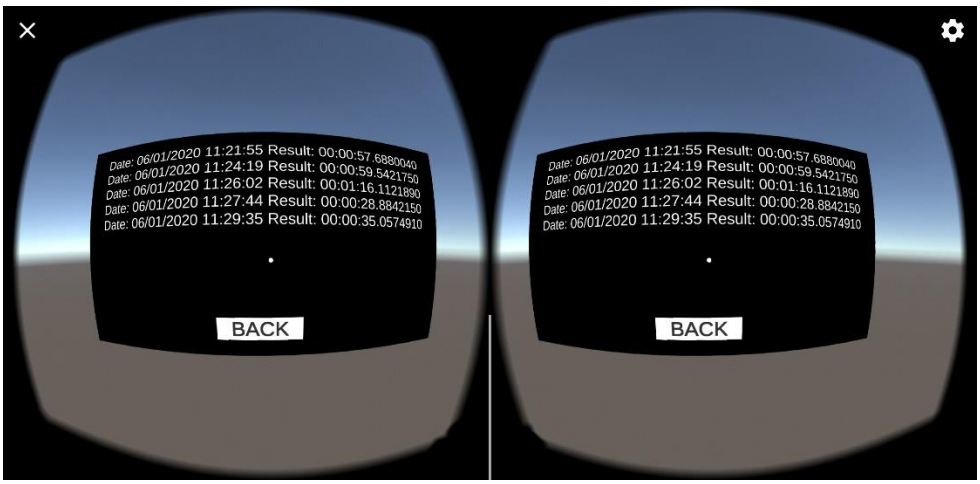


Рисунок 4.4. UI результатів часу проходження лабіринту.

На рис. 4.4 ми можемо побачити UI дати, коли користувач отримав результати часу проходження лабіринту та самі результати проходження. Всі ці результати зберігається у сховищі, завдяки чому їх можна переглядати будь-коли, вони будуть збережені, навіть, якщо користувач вийде із системи та зайде в неї через будь-який проміжок часу.

Цей UI корисний для перевірки своїх результатів, користувач може бачити покращує він свої результати проходження чи ні.

4.3 Взаємодія зі сценою з першим варіантом переміщення

На рис. 4.5 ми можемо бачити вид, який побачить користувач після заповнення Progress Bar інтерактивного об’єкту головного меню Maze move with teleports і після цього користувач може почати проходити лабіринт.

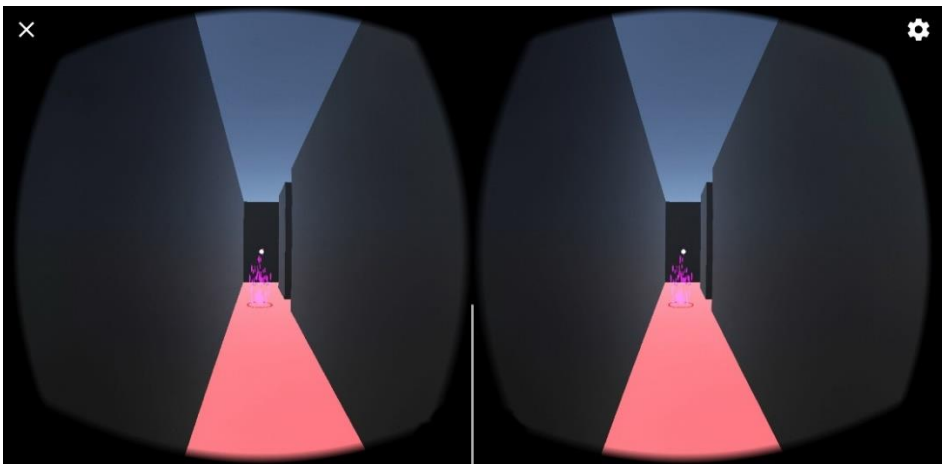


Рисунок 4.5. Сцена першого варіанту переміщення.

Після наведення погляду на телепорт відбувається переміщення користувача на позицію телепорту та збільшується значення точки, яка відображається на UI, який знаходиться над користувачем. Всі дані з цього UI зберігаються у іншому UI з назвою POINTS (CURRENT RESULT), який можна відкрити за допомогою інтерактивного об’єкту у UI з картою. Дані зберігаються у сховищі, а тому останній результат точок буде зберігатися, навіть, якщо користувач вийде з гри и зайде в неї знов.

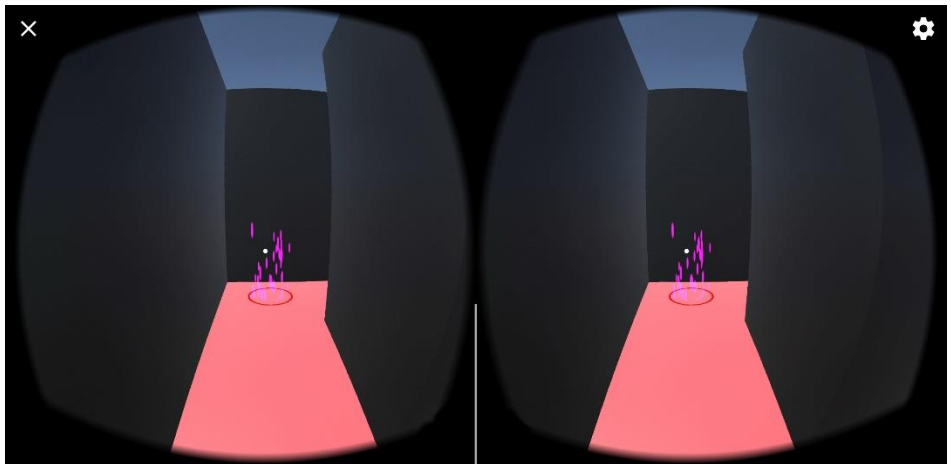


Рисунок 4.6. Вид після наведення погляду на передній телепорт.

На рис. 4.6 ми бачимо що відбудеться після наведення погляду користувача на найближчий передній телепорт. Ми бачимо що він перемістився на місце цього телепорту і може продовжувати своє переміщення по лабіринту далі, наводячи погляд на інші телепорти.

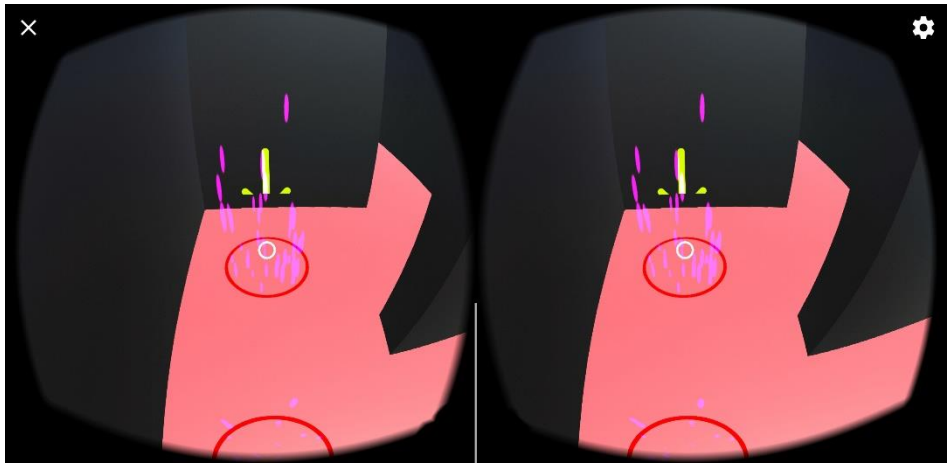


Рисунок 4.7. Процес заповнення Progress Bar (стрілка вниз).

На рисунку 4.7 ми бачимо процес заповнення Progress Bar, після наведення погляду користувача на телепорт.

У цьому випадку стрілка, яка вказує вниз, виступає як Progress Bar. Після заповнення стрілки, спрямованої униз, користувач переміститься на місце позиції точки телепорту.

Аналіз підрахунку точок переміщення

Якщо користувач подивиться вгору, то побачить два UI. Один з підрахунком точок переміщення, другий для відеозапису проходження.

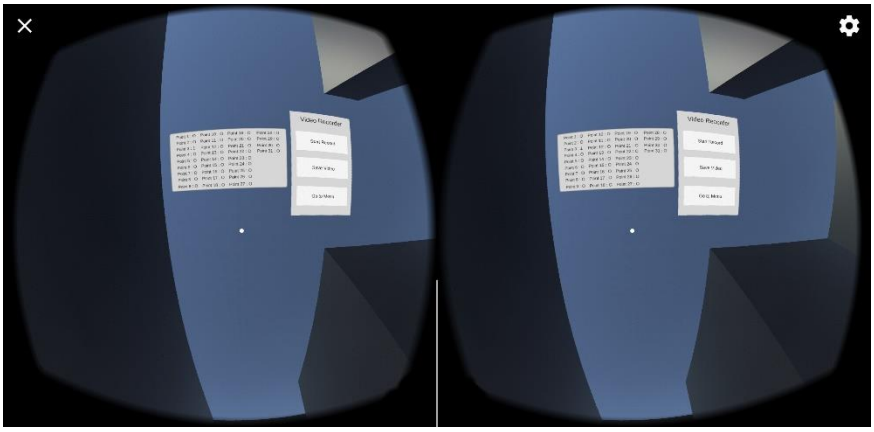


Рисунок 4.8. Відображення двох UI.

На рис. 4.8 можна побачити, що після переміщення користувача на передній телепорт, який виступає у якості точки під номером 3, точка збільшилася на одиницю, тобто відбулася операція інкремент. І кожного разу, коли користувач буде перебувати на місці цієї точки, вона буде збільшуватися на одиницю. Тобто, якщо користувач за час проходження лабіринту буде присутній на цій точці 3 рази, то значення її буде рівне трьом. Це дає можливість відстежувати рух користувача по віртуальному лабіринту.

В головному меню додатку, користувач має можливість відкрити карту з номерами точок переміщення, які розміщенні по лабіринту. Це дає можливість перевірити де саме і скільки разів користувач був присутнім під час проходження лабіринту.

Аналіз відеозапису дій користувача

Поряд з UI підрахунку точок переміщення знаходиться UI для відеозапису проходження (Рисунок 4.8). Якщо користувач наведе погляд на інтерактивний об’єкт з назвою “Start Record”, то з’явиться надпис з проханням підтвердити відеозапис користувачу (Рисунок 4.9).

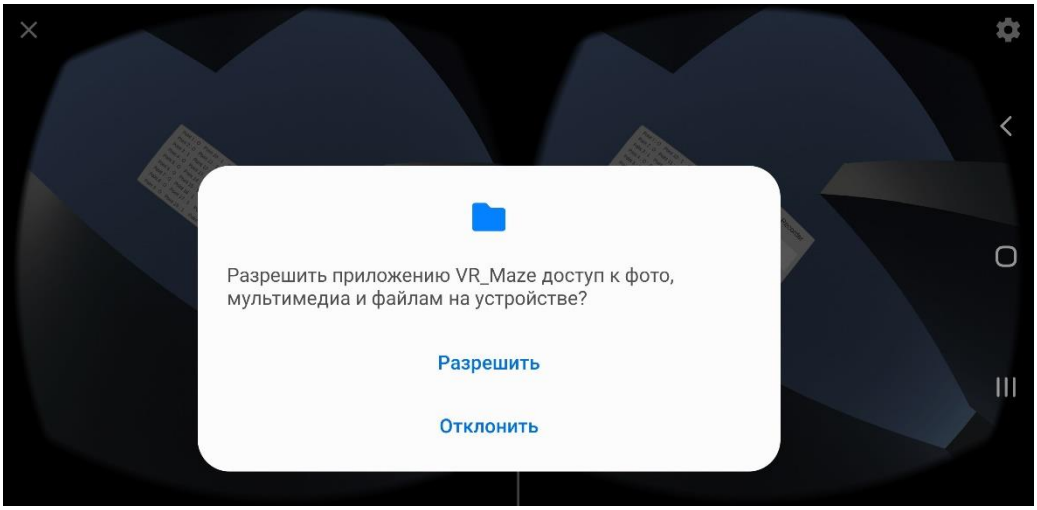


Рисунок 4.9. Надпис з проханням для підтвердження відеозапису.

Після погодження почнеться відеозапис всіх дій користувача. Для того, щоб користувач міг зберегти відеозапис у галерею на своєму мобільному пристрої, йому необхідно навести погляд на інтерактивний об’єкт з назвою “Save Video” і відеозапис буде збережено у галерею користувача (Рисунок 4.10).

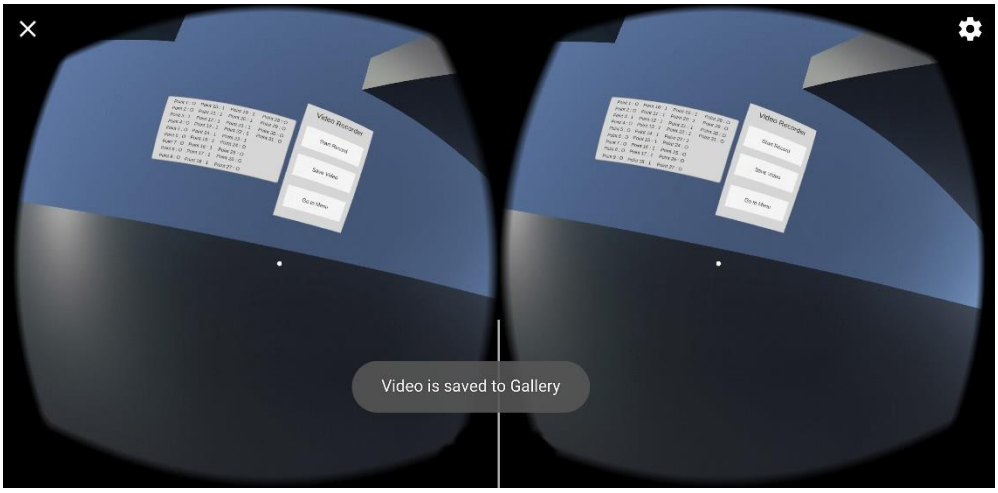


Рисунок 4.10. Надпис повідомлення, що відеозапис збережено.

На рис. 4.10 ми можемо бачити надпис повідомлення користувачу, що відеозапис було збережено у галерею користувача.

Після збереження відеозапису у галерею користувачу, цей відеозапис може бути переглянутий для подальшого аналізу.

Аналіз проходження лабіринту

Коли користувач знайде вихід із віртуального лабіринту, то він побачить надпис “Exit”, а для завершення проходження йому необхідно навести погляд на ручку, яка знаходиться нижче надпису.

Після наведення погляду на ручку та заповнення Progress Bar, відбудеться анімація появи UI з вітанням о проходженні лабіринту, з часом проходження користувача, з написом позитивний чи негативний результат проходження відносно попереднього результату проходження та з написом о повторному проходженні лабіринту (Рисунок 4.12).

Час о проходженні виводиться для того, щоб користувач міг зрозуміти за скільки часу йому вдалося знайти вихід із лабіринту. Кожного разу після проходження лабіринту, користувач може порівнювати свій час проходження з попереднім часом і бачити покращує він свій результат чи ні.

Якщо користувач наведе погляд на об’єкт Restart the Maze, то сцена перезавантажиться і користувач з самого початку розпочне своє проходження.

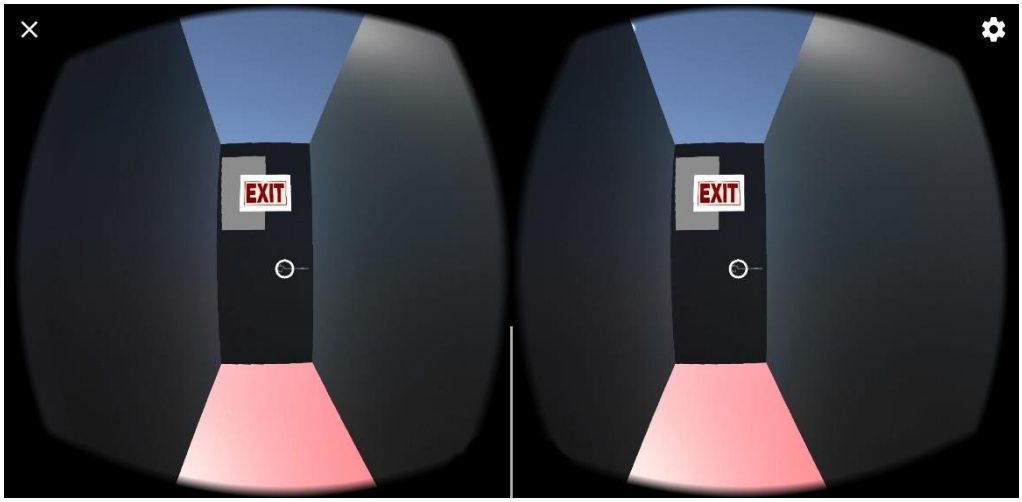


Рисунок 4.11. UI виходу з процесом заповнення Progress Bar.

На рис. 4.11 можна побачити процес заповнення Progress Bar, коли користувач навів погляд на інтерактивний об’єкт ручку.

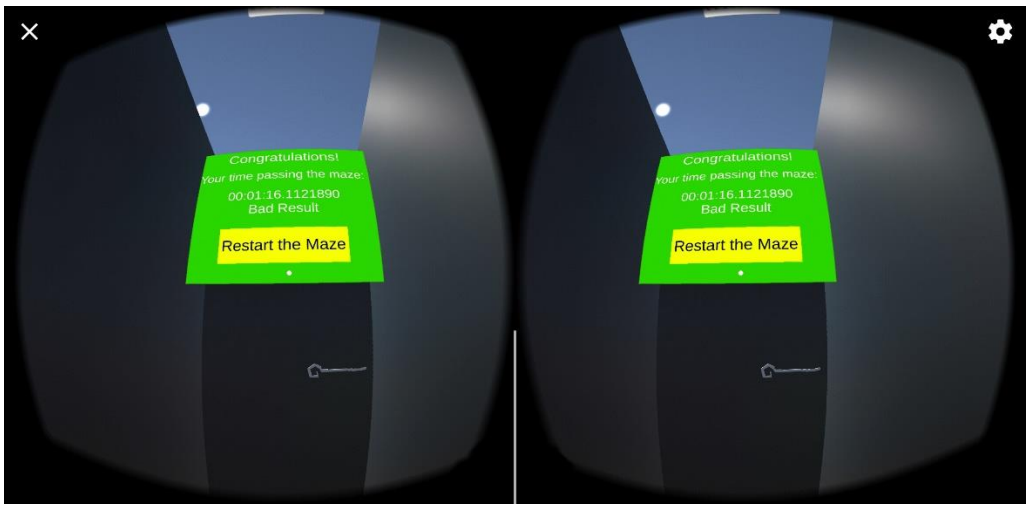


Рисунок 4.12. UI з результатом часу проходження лабіринту.

На рис. 4.12 ми бачимо результат часу проходження користувача віртуального лабіринту, який складає 1 хвилину, 16 секунд та 112 мілісекунди.

Також можемо побачити, що користувач отримав результат Bad Result, це означає, що попереднього разу користувач пройшов лабіринт швидше ніж цього разу. У системі постійно зберігається результат часу останнього проходження лабіринту користувачем. Цей час зберігається у сховищі, а отже буде збереженим, навіть, якщо користувач вийде з системи та зайде в неї через декілька днів. Цей збережений час постійно порівнюється з поточним часом проходження користувача. Якщо поточний час проходження користувача буде меншим за попередній час проходження, то користувачу відобразиться надпис Good Result.

4.4 Взаємодія з другим варіантом переміщення

Коли користувач наведе свій погляд на інтерактивний об’єкт головного меню Maze move with arrows і дочекається заповнення Progress Bar, то йому відріється сцена віртуального лабіринту з другим варіантом переміщення.

Коли користувач опустить голову униз, то побачить UI зі стрілками для пересування, а коли наведе погляд на одну із стрілок, то почне пересуватися по віртуальному лабіринту у напрямку цієї стрілки.

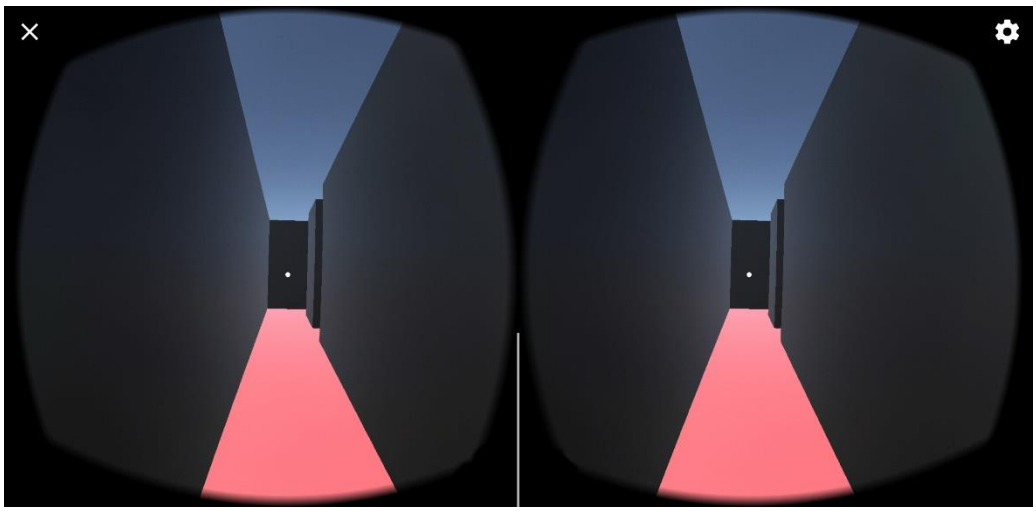


Рисунок 4.13. Вид сцени з другим варіантом переміщення.

На рис. 4.13 можна побачити вид, який побачить користувач, коли відкриє сцену з другим варіантом переміщення. У цій сцені вже немає телепортів, а є UI зі стрілками для переміщення користувача, який знаходиться внизу під користувачем. Цей UI є статичним, тому він переміщається постійно разом з переміщенням самого користувача. Також, UI з підрахунком точок переміщення та UI для відеозапису проходження також є статичними та постійно переміщаються разом із користувачем, тому користувач постійно може бачити всі ці UI під час проходження лабіринту.

UI з обчисленням точок переміщення такий же, як і у першому варіанті переміщення. Всі точки розташовані, як і у першому варіанті та їх можна звіряти з картою, яка знаходиться у головному меню. UI для відеозапису працює так само як і у першому варіанті переміщення. UI з виходом із лабіринту в цьому варіанті переміщення працює так само як і у першому варіанті переміщення.

4.5 Взаємодія з грою для тренування пам’яті

Коли користувач в головному меню системи наведе погляд на інтерактивний об’єкт з назвою Memory Game та дочекається заповнення Progress Bar, то йому відриється UI з вибором варіанту гри для тренування пам’яті, де користувач може обрати гру на 4, 8 та 12 карток. Для цього

користувачу також потрібно навести погляд на цифру з кількістю карток, з якою він хоче грати та знову дочекатися заповнення Progress Bar, після заповнення відкриється сцена з грою для тренування пам’яті.

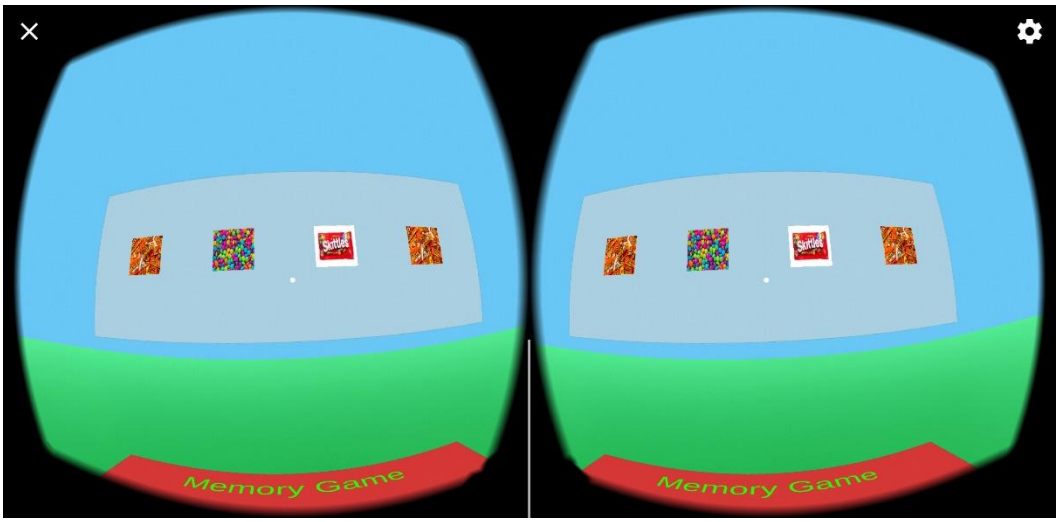


Рисунок 4.14. UI гри для тренування пам’яті на 4 картки.

На рис. 4.14 можна побачити гру для тренування пам’яті користувача на 4 картки. Для відкриття картки, користувачу необхідно навести погляд на одну з карток та дочекатися заповнення Progress Bar.

Коли користувач знайде усі пари карток, то гра закінчиться і відобразиться UI з результатом гри та інтерактивним об’єктом з назвою Try Again.

Результат гри демонструє кількість разів, скільки користувач відкривав пари карток, щоб їх відгадати. Завдяки такому результату користувач може постійно бачити чи покращує він свій результат гри чи ні.

Коли користувач наведе свій погляд на інтерактивний об’єкт Try Again та дочекається заповнення Progress Bar, тоді гра перезапуститься і користувач зможе зіграти в неї знову.

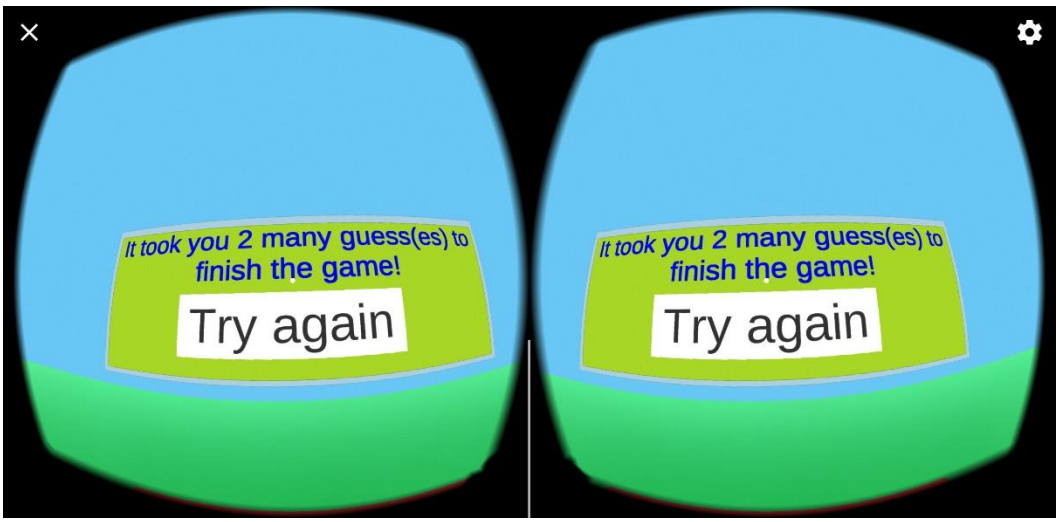


Рисунок 4.15. UI з результатом гри на пам'ять для 4 карток.

На рис. 4.15 можемо бачити, що користувачу знадобилося 2 спроби, щоб відкрити картки та знайти для них пару та завершити гру.

У будь-який момент гри, користувач може подивитися вниз та побачити там UI з інтерактивним об’єктом Go to Menu (Рисунок 4.16).

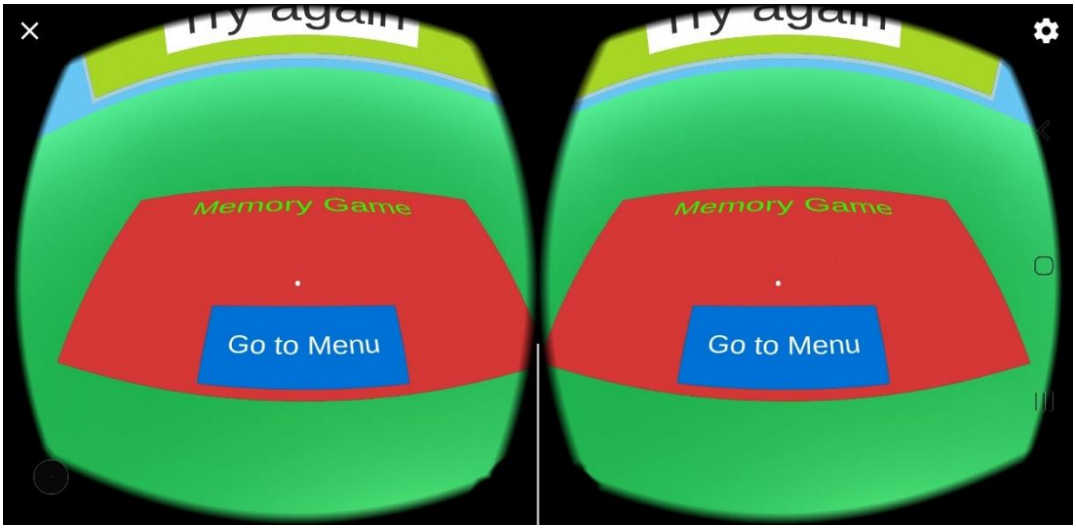


Рисунок 4.16. UI для виходу з гри для тренування пам’яті.

На рис. 4.16 можна побачити UI з інтерактивним об’єктом Go to Menu. Коли користувач наведе свій погляд на цей об’єкт та дочекається заповнення Progress Bar, він повернеться у головне меню системи, де зможе обрати інший варіант гри для тренування пам’яті.

ВИСНОВОК ДО РОЗДІЛУ 4

У четвертому розділі був проведений аналіз розробленої системи.

Була розглянута взаємодія користувача із системою. Система була перевірена на смартфоні Samsung Galaxy S9+ з OS Android з версією 10.1. Був проведений аналіз головного меню системи.

Була детально розглянута взаємодія користувача з головним меню. Була розглянута карта лабіринту з точками переміщення, яку користувач може використовувати для розуміння де та скільки разів він знаходився під час проходження віртуального лабіринту.

Була розглянута взаємодія користувача зі сценою з першим варіантом переміщення. Були наведені скріншоти, на яких можна побачити вид, який бачить користувач під час проходження лабіринту з першим варіантом переміщення. Був проведений аналіз підрахунку точок переміщення та був наведений скріншот з гри, який демонструє коректність підрахунку точок. Був проведений аналіз відеозапису дій користувача. Були представлені скріншоти, які демонструють прохання підтвердити відеозапис користувача та збереження відео у галерею користувача. Був проведений аналіз проходження лабіринту, а саме, що побачить користувач, коли знайде вихід. Був наведений скріншот, який демонструє, що відбудеться після наведення погляду користувача на ручку та заповнення Progress Bar. Був наведений скріншот UI результату, який показує час за який користувач зміг знайти вихід із віртуального лабіринту.

Була розглянута взаємодія користувача з другим варіантом переміщення. Були показані скріншоти з видом, який бачить користувач, коли потрапить у лабіринт з другим варіантом переміщення.

Була розглянута взаємодія користувача з грою для тренування пам’яті. Був наведений скріншот на якому видно UI гри для тренування пам’яті, яку грав користувач. Був наведений скріншот з результатом гри для тренування пам’яті. Всі наведені скріншоти у розділі демонструють коректність та правильність розробленої системи.

ВИСНОВКИ

У даній роботі створено систему віртуальної реальності для визначення дій людини у лабіринті, яка занурює користувача у віртуальний простір та відстежує як користувач в ньому орієнтується.

У першому розділі була детально розглянута технологія віртуальної реальності, її історія, ключові поняття та тенденції розвитку. Детально були розглянуті пристрої для взаємодії з віртуальною реальністю. Була наведена таблиця існуючих систем віртуальної реальності для допомоги людям у різних сферах життя та був зроблений аналіз порівняння цих систем. Було наведено ряд переваг моєї системи.

У другому розділі розглядались технології для розробки системи. Було зроблено пояснення чому саме такі технології були обрані для створення системи, в чому їх переваги та як дані технології допомагають розробникам під час створення системи віртуальної реальності.

У третьому розділі було розглянуто деталі розробки системи. Які сцени та об’єкти були створенні. Як було реалізовано перехід між сценами. Як було реалізовано переміщення користувача по віртуальному лабіринту. Як було реалізовано збір інформації під час проходження лабіринту. Як була реалізована можливість відеозапису та як була реалізована гра для тренування пам’яті. Були розглянуті написанні класи та їх методи, які використовувались для реалізації системи.

У четвертому розділі був проведений аналіз розробленої системи. Перевірка відбувалася на смартфоні Samsung Galaxy S9+ з OS Android 10.1. Був зроблений тест відкриття всіх сцен, два варіанта переміщення по лабіринту, коректність збору та збереження даних та гри для тренування пам’яті.

Під час розробки системи були враховані всі вимоги, які були зазначені у плані даної дипломної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Психіатрія і наркологія”. За редакцією професора В.Л. Гавенка, чл.-кор. АМН України, професора В.С. Бітенського. ст. 245-249.

2. Віртуальна реальність [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/ru/knowledge-base/technology-innovation/virtualnaja-realnost-vr (дата звернення 12.05.2020).

3. Samsung Gear VR [Електронний ресурс] – Режим доступу до ресурсу: https://www.gift-ideas-finder.com/product/samsung-gear-vr-virtual-reality-headset/ (дата звернення 12.05.2020).

4. Доповнена реалність (AR) [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/knowledge-base/technology-innovation/dopolnennaja-realnost-ar (дата звернення 12.05.2020).

5. Big Bang AR [Електронний ресурс] – Режим доступу до ресурсу: https://shazoo.ru/2019/03/07/76833/prilozhenie-big-bang-ar-pozvolyaet-uvidet-istoriyu-vselennoj-v-dopolnennoj-realnosti (дата звернення 12.05.2020).

6. Microsoft mixed reality [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtudo.com.br/noticias/noticia/2015/02/como-a-microsoft-pretende-popularizar-o-uso-dos-hologramas-em-2015.html (дата звернення 12.05.2020).

7. Історію віртуальної реальності [Електронний ресурс] – Режим доступу до ресурсу: https://bit.ly/2yTceWQ (дата звернення 14.05.2020).

8. Pilot training [Електронний ресурс] – Режим доступу до ресурсу: https://www.maxwell.af.mil/News/Photos/igphoto/2001864957/ (дата звернення 14.05.2020).

9. Oculus Rift [Електронний ресурс] – Режим доступу до ресурсу: https://cyber1.xda.uz/virtualnaya-realnost-vr/oculus-rift-s (дата звернення 14.05.2020).

10. Шолом ВР [Електронний ресурс] – Режим доступу до ресурсу: bhttps://bit.ly/2Bdplmh (дата звернення 14.05.2020).
11. Віртуальна реальність [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/ru/knowledge-base/technology-innovation/virtualnaja-realnost-vr (дата звернення 14.05.2020).
12. Перспективи розвитку ВР [Електронний ресурс] – Режим доступу до ресурсу: https://c.mi.com/thread-1045686-1-0.html (дата звернення 14.05.2020).
13. Вірутальна реальність [Електронний ресурс] – Режим доступу до ресурсу: https://bit.ly/2AsoEFC (дата звернення 14.05.2020).
14. Дроти у ВР [Електронний ресурс] – Режим доступу до ресурсу: https://igate.com.ua/lenta/11102-provoda-meshayut-prodvizheniyu-ustrojstv-virtualnoj-realnosti (дата звернення 14.05.2020).
15. Tracking [Електронний ресурс] – Режим доступу до ресурсу: https://venturebeat.com/2019/05/05/how-virtual-reality-positional-tracking-works/ (дата звернення 14.05.2020).
16. Motion Controllers [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Virtual_reality (дата звернення 14.05.2020).
17. How virtual and augmented reality enhance assembly training? [Електронний ресурс] – Режим доступу до ресурсу: https://giantlazer.com/how-virtual-and-augmented-reality-enhance-assembly-training/ (дата звернення 14.05.2020).
18. Eye tracking in VR [Електронний ресурс] – Режим доступу до ресурсу: https://www.worldviz.com/virtual-reality-eye-tracking-for-research-solutions (дата звернення 14.05.2020).
19. Eye tracking in practice [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@ergomania_UX/eye-tracking-in-practice-64724c0a5f64 (дата звернення 14.05.2020).

20. Foveated Rendering [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Foveated_rendering (дата звернення 14.05.2020).
21. Foveated rendering [Електронний ресурс] – Режим доступу до ресурсу: https://onebonsai.com/blog/how-eye-tracking-will-improve-digital-communication/ (дата звернення 14.05.2020).
22. Killer app [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Killer_application (дата звернення 14.05.2020)
23. Віртуальна реальність [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/ru/articles/virtualnaja-realnost-vr-luchshie-praktiki (дата звернення 14.05.2020).
24. Інвестування у ВР [Електронний ресурс] – Режим доступу до ресурсу: https://ru.ihodl.com/investment/2016-03-22/11-sposobov-investirovat-v-virtualnuiu-realnost/ (дата звернення 14.05.2020).
25. Пристрої ВР [Електронний ресурс] – Режим доступу до ресурсу: https://hub.packtpub.com/top-7-modern-virtual-reality-hardware-systems/ (дата звернення 14.05.2020).
26. Ринок ВР [Електронний ресурс] – Режим доступу до ресурсу: https://bit.ly/2MhdOoi (дата звернення 15.05.2020).
27. HoloLens 2 [Електронний ресурс] – Режим доступу до ресурсу: https://www.reddit.com/r/AR_MR_XR/comments/g2o1nr/microsoft_finally_expands_the_availability_of/ (дата звернення 15.05.2020).
28. Interaction with user and computer application. [Електронний ресурс] – Режим доступу до ресурсу: https://kazan.postupi.online/vuz/kfu/programma-magistr/9344/ (дата звернення 15.05.2020).
29. Interaction with user and virtual glasses [Електронний ресурс] – Режим доступу до ресурсу: https://www.oppermann-events.de/en/virtual-and-

augmented-reality-for-trade-fairs-and-events/virtual-reality-glasses-rental.html (дата звернення 15.05.2020).
30. Імерсивні технології [Електронний ресурс] – Режим доступу до ресурсу: https://bit.ly/3dkVZ3A (дата звернення 15.05.2020).
31. Імерсивність [Електронний ресурс] – Режим доступу до ресурсу: https://bit.ly/2ZNVVWi (дата звернення 15.05.2020).
32. Google Cardboard [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Google_Cardboard (дата звернення 15.05.2020).
33. Google Cardboard [Електронний ресурс] – Режим доступу до ресурсу: https://store.google.com/CA/product/google_cardboard (дата звернення 15.05.2020).
34. Samsung Gear VR [Електронний ресурс] – Режим доступу до ресурсу: https://www.desertcart.com.kw/products/46238180-samsung-gear-vr-w-controller-2017-sm-r-325-nzvaxar-us-version-w-warranty (дата звернення 15.05.2020).
35. Samsung Gear VR [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Samsung_Gear_VR (дата звернення 15.05.2020).
36. Oculus Rift [Електронний ресурс] – Режим доступу до ресурсу: https://www.bestadvisers.co.uk/virtual-reality-headsets/hp-windows-mixed-reality-vs-oculus-rift-s (дата звернення 15.05.2020).
37. Oculus VR [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Oculus_VR (дата звернення 15.05.2020).
38. HTC Vive [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/HTC_Vive (дата звернення 15.05.2020).
39. HTC Vive [Електронний ресурс] – Режим доступу до ресурсу: https://www.bestbuy.com/site/htc-vive-virtual-reality-system-for-compatible-windows-pcs-black/5901551.p?skuId=5901551 (дата звернення 15.05.2020).

40. PS VR [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PlayStation_VR (дата звернення 15.05.2020).
41. PlayStation VR [Електронний ресурс] – Режим доступу до ресурсу: https://it.ign.com/playstation-vr-project-morpheus/126861/news/playstation-vr-tutti-i-giochi-annunciati-alla-conferenza (дата звернення 15.05.2020).
42. Valve Index [Електронний ресурс] – Режим доступу до ресурсу: https://picclick.it/Valve-Index-VR-Kit-Index-Headset-Index-Controller-254339444947.html (дата звернення 18.05.2020).
43. Valve Index [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Valve_Index (дата звернення 18.05.2020).
44. Process of Interaction with interactive object [Електронний ресурс] – Режим доступу до ресурсу: https://www.androidpolice.com/2015/11/04/cardboard-v1-7-adds-a-slick-welcome-video-and-new-demo-screen-photospheres-for-app-screenshots-drops-windy-day-apk-download/ (дата звернення 18.05.2020).
45. Prototyping System [Електронний ресурс] – Режим доступу до ресурсу: https://www.techdesignforums.com/practice/guides/system-virtual-prototyping/ (дата звернення 18.05.2020).
46. The Body VR [Електронний ресурс] – Режим доступу до ресурсу: https://xinreality.com/wiki/The_Body_VR (дата звернення 18.05.2020).
47. Tilt Brush [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Tilt_Brush (дата звернення 18.05.2020).
48. Sea Hero Quest [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Sea_Hero_Quest (дата звернення 18.05.2020).
49. Virtual reality maze 'predicts Alzheimer's disease' [Електронний ресурс] – Режим доступу до ресурсу: BBC - https://www.bbc.com/news/health-34607267 (дата звернення 18.05.2020).

50. Morganti, Francesca, Stefano Stefanini, and Giuseppe Riva. "From allo-to egocentric spatial ability in early Alzheimer's disease: a study with virtual reality spatial tasks." Cognitive neuroscience 4.3-4 (2013): 171-180.
51. Sounding the alarm on a future epidemic: Alzheimer's disease. [Електронний ресурс] – Режим доступу до ресурсу: https://newsroom.ucla.edu/stories/sounding-the-alarm-on-a-future-epidemic:-alzheimer-s-disease (дата звернення 18.05.2020).
52. ALZ [Електронний ресурс] – Режим доступу до ресурсу: https://www.alz.co.uk/research/statistics (дата звернення 18.05.2020).
53. Google Daydream [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalbodies.net/360-video/ (дата звернення 18.05.2020).
54. Glove VR [Електронний ресурс] – Режим доступу до ресурсу: https://newatlas.com/vr-touch-resistance-gloves/49790/ (дата звернення 18.05.2020).
55. Leap Motion [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Leap_Motion (дата звернення 18.05.2020).
56. Leap motion VR [Електронний ресурс] – Режим доступу до ресурсу: https://leap.quitebeyond.de/leap-motion-vr-bundle-unboxing-weightless-gameplay-oculus-rift-3/ (дата звернення 18.05.2020).
57. Unity Engine [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Unity_(game_engine) (дата звернення 15.05.2020).
58. Linux in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://blogs.unity3d.com/2019/05/30/announcing-the-unity-editor-for-linux/ (дата звернення 15.05.2020).
59. Scene in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/560/Documentation/Manual/CreatingScenes.html (дата звернення 15.05.2020).

60. Move in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/SceneViewNavigation.html (дата звернення 15.05.2020).
61. Positioning GameObjects in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/PositioningGameObjects.html (дата звернення 15.05.2020).
62. GameView [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/GameView.html (дата звернення 15.05.2020).
63. Inspector in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/UsingTheInspector.html (дата звернення 15.05.2020).
64. Hierarchy in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/Hierarchy.html (дата звернення 15.05.2020).
65. Project View in Unity. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/ProjectView.html (дата звернення 15.05.2020).
66. Console in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/Console.html. (дата звернення 15.05.2020)
67. Animator Window in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/AnimatorWindow.html (дата звернення 15.05.2020).
68. About Animation Editor in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/animeditor-UsingAnimationEditor.html (дата звернення 15.05.2020).

69. Ігровий рушій Unity [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unity_(рушій_гри) (дата звернення 15.05.2020).
70. VR and AR in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/features/multiplatform (дата звернення 15.05.2020).
71. VR in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/unity/features/vr (дата звернення 15.05.2020).
72. VR and AR in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/solutions/ar-and-vr-games (дата звернення 15.05.2020).
73. Google VR SDK [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/vr/develop/unity/get-started-android (дата звернення 15.05.2020).
74. Prefab in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/Prefabs.html (дата звернення 15.05.2020).
75. Plugin in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/Plugins.html (дата звернення 15.05.2020).
76. Scripting in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/ScriptingSection.html (дата звернення 15.05.2020).
77. C#, .Net, .NET Framework [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@nirajv21/introduction-to-c-net-and-mono-76b7ddad1788 (дата звернення 15.05.2020).
78. How Unity Supports Cross Platform Feature [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@nirajv21/how-unity-supports-cross-platform-feature-ae722321cfa (дата звернення 15.05.2020).

79. .NET Profile in Unity. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/dotnetProfileSupport.html (дата звернення 15.05.2020).
80. Grid Layout Group [Електронний ресурс] – Режим доступу до https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/script-GridLayoutGroup.html (дата звернення 18.05.2020).
81. Particle system [Електронний ресурс] – Режим доступу до https://docs.unity3d.com/Manual/ParticleSystems.html (дата звернення 15.05.2020).
82. Rigid Body [Електронний ресурс] – Режим доступу до https://docs.unity3d.com/Manual/class-Rigidbody.html (дата звернення 15.05.2020).
83. PlayerPrefs [Електронний ресурс] – Режим доступу до https://docs.unity3d.com/ScriptReference/PlayerPrefs.html (дата звернення 15.05.2020).
84. UI.Image.fillAmount in Unity[Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/540/Documentation/ScriptReference/UI.Image-fillAmount.html (дата звернення 15.05.2020).
85. Transform.position in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/ScriptReference/Transform-position.html (дата звернення 18.05.2020).
86. Mathf.Lerp in Unity. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/ScriptReference/Mathf.Lerp.html (дата звернення 15.05.2020).
87. Coroutine [Електронний ресурс] – Режим доступу до https://docs.unity3d.com/ScriptReference/Coroutine.html (дата звернення 18.05.2020).

88. Transform.Translate in Unity. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/ScriptReference/Transform.Translate.html (дата звернення 15.05.2020).
89. Time.deltaTime in Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/ScriptReference/Time-deltaTime.html (дата звернення 15.05.2020).
90. Canvas in Unity. [Електронний ресурс] – Режим доступу до https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UICanvas.html (дата звернення 15.05.2020).

ДОДАТОК 1

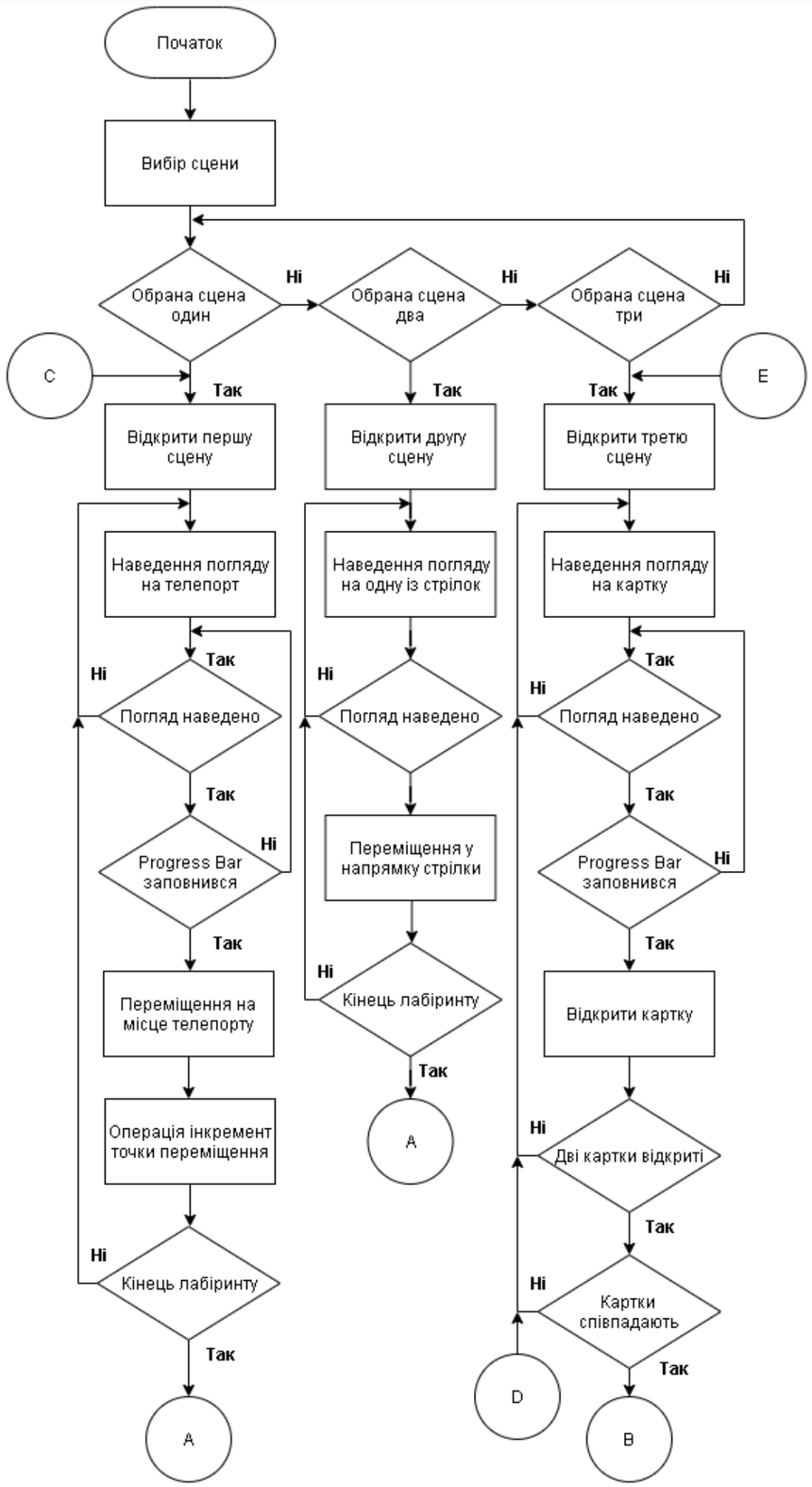
Система віртуальної реальності для визначення дій людини у лабіринті

Алгоритм дій програмного забезпечення

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2020 р



					ІАЛЦ.467200.004 Д1	
					Система віртуальної реальності для визначення дій людини у лабіринті Алгоритм дій програмного забезпечення	Літера / Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Каленик П.І.				Аркуш 1 / Аркушів 1
Перевірив		Волокита А.М.				
Н.контр.		Сімоненко В.П.			Дипломна робота	НТУУ "КПІ Ім Ігоря Сікорського", ФІОТ, гр. ІП-64
Затверд.						

ДОДАТОК 2

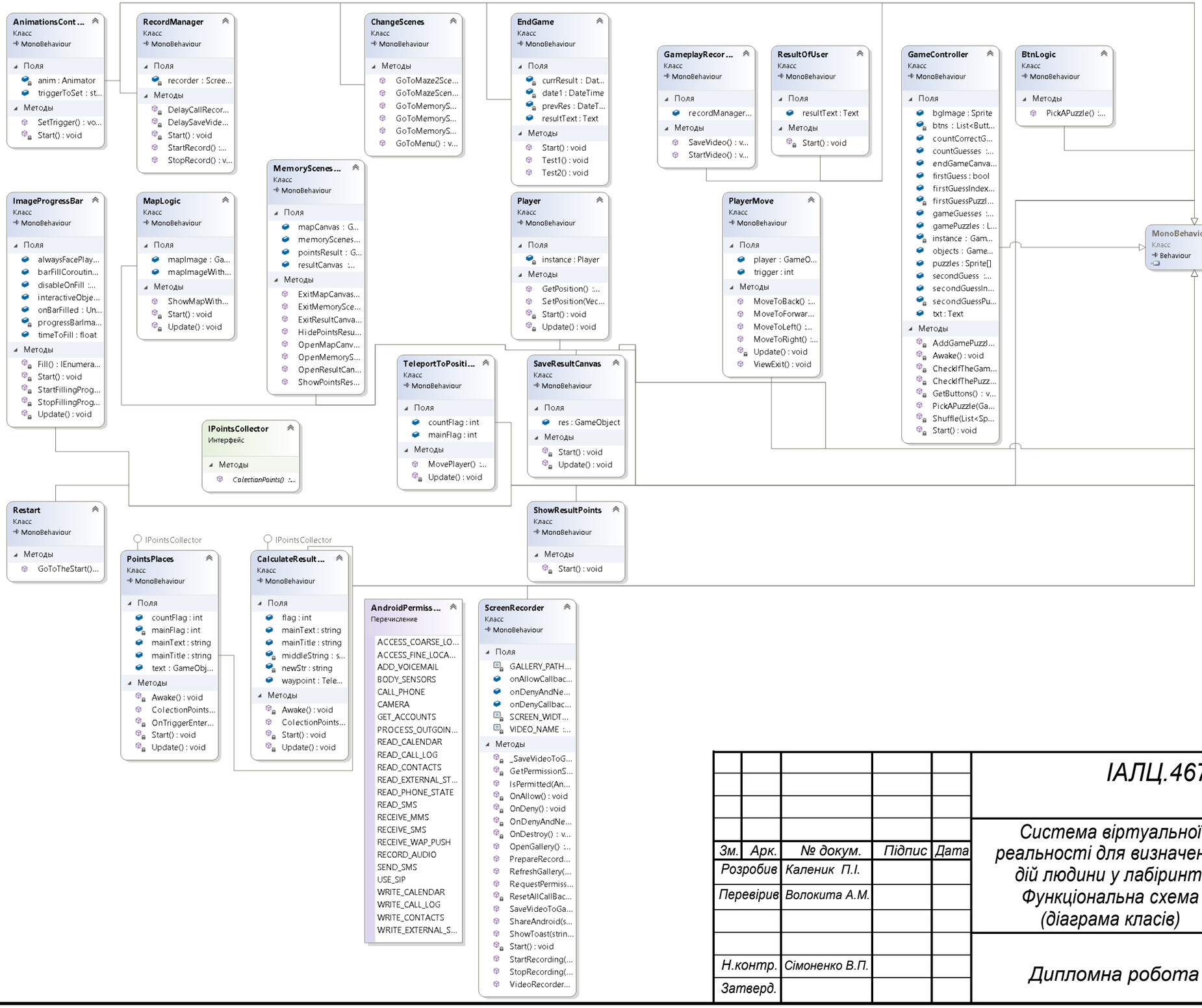
Система віртуальної реальності для визначення дій людини у лабіринті

Функціональна схема (діаграма класів)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2020 р



					ІАЛЦ.467200.005 Д2			
					Система віртуальної реальності для визначення дій людини у лабіринті Функціональна схема (діаграма класів)	Літера	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Каленик П.І.				Аркуш 1	Аркушів 1	
Перевірив		Волокита А.М.						
					Дипломна робота	НТУУ "КПІ Ім Ігоря Сікорського", ФІОТ, гр. ІП-64		
Н.контр.		Сімоненко В.П.						
Затверд.								

ДОДАТОК 3

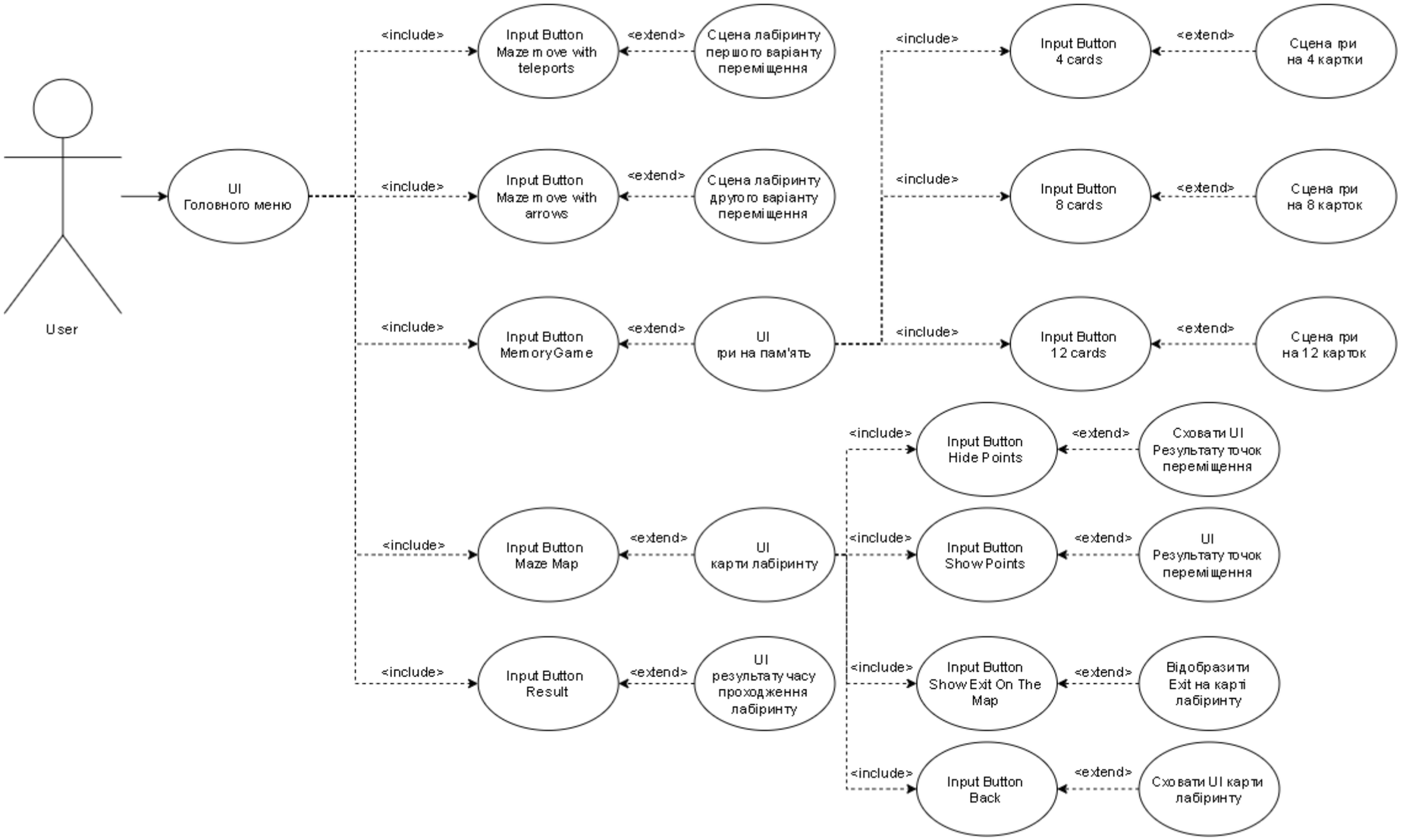
Система віртуальної реальності для визначення дій людини у лабіринті

Структурна схема системи

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2020 р



					ІАЛЦ.467200.006 ДЗ	
					Система віртуальної реальності для визначення дій людини у лабіринті Структурна схема системи	Літера / Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Каленик П.І.				Аркуш 1 / Аркушів 1
Перевірив		Волокита А.М.				
.					Дипломна робота	НТУУ "КПІ Ім Ігоря Сікорського", ФІОТ, гр. ІП-64
Н.контр.		Сімоненко В.П.				
Затверд.						

ДОДАТОК 4

Система віртуальної реальності для визначення дій людини у лабіринті

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 13

Київ 2020 р

Клас CalculateResultPoints

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class CalculateResultPoints : MonoBehaviour
{
   public TeleportToPosition waypoint;
   private string middleString;
   string newStr;
   public int flag;
   public string mainText;
   public string mainTitle;
   private void Awake()
   {
      gameObject.GetComponent<Text>().text += "O";
   }
   private void Start()
   {
      mainText = waypoint.countFlag.ToString();
      mainTitle = gameObject.GetComponent<Text>().text;
      PlayerPrefs.SetString($"{gameObject.name}", "");
   }
   private void Update()
   {
      if (mainText == waypoint.countFlag.ToString())
      {
         Debug.Log("Equel");
      }
      else
      {
         mainTitle = mainTitle.Replace("O", "");
         mainText = waypoint.countFlag.ToString();
         gameObject.GetComponent<Text>().text = mainTitle + mainText;
         PlayerPrefs.SetString($"{gameObject.name}", mainText);
         Debug.Log(gameObject.name + PlayerPrefs.GetString($"{gameObject.name}"));
      } } }
```

Клас Player

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour
{
   private static Player instance;

   private void Start()
   {
```

```
        instance = this;
    }

    private void Update()
    {
        transform.rotation = Quaternion.Euler(0, -90, 0);
        transform.position = new Vector3(transform.position.x, 1.8f, transform.position.z);
    }

    public static Vector3 GetPosition()
    {
        return instance.transform.position;
    }

    public static void SetPosition(Vector3 position)
    {
        instance.transform.position = new Vector3(position.x, 1.8f, position.z);
    }
}
```

Клас PlayerMove

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Experimental.PlayerLoop;
public class PlayerMove : MonoBehaviour
{
    public GameObject player;

    public int trigger = 0;
    private void Update()
    {
        if (trigger == 1)
        {
            player.transform.Translate(Vector3.forward * Time.deltaTime);
        }
        else if (trigger == 2)
        {
            player.transform.Translate(Vector3.right * Time.deltaTime);
        }
        else if (trigger == 3)
        {
            player.transform.Translate(Vector3.back * Time.deltaTime);
        }
        else if (trigger == 4)
        {
            player.transform.Translate(Vector3.left * Time.deltaTime);
        }
    }
```

```
    public void MoveToForward()
    {
        trigger = 1;
        Debug.Log("Forward");
    }
    public void MoveToRight()
    {
        trigger = 2;
        Debug.Log("Right");
    }
    public void MoveToBack()
    {
        trigger = 3;
        Debug.Log("Back");
    }
    public void MoveToLeft()
    {
        trigger = 4;
        Debug.Log("Left");
    }
    public void ViewExit()
    {
        trigger = 0;
    }
}
```

Клас PointsPlaces

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class PointsPlaces : MonoBehaviour
{
    private static int mainFlag;
    public int countFlag;
    public string mainText;
    public string mainTitle;
    public GameObject text;

    private void Awake()
    {
        text.GetComponent<Text>().text += "O";
    }
    private void Start()
    {
        mainText = countFlag.ToString();
        mainTitle = text.GetComponent<Text>().text;
```

```
        PlayerPrefs.SetString($"{gameObject.name}", "");
    }

    private void Update()
    {
        if (mainText == countFlag.ToString())
        {
            Debug.Log("Equel");
        }
        else
        {
            mainTitle = mainTitle.Replace("O", "");
            mainText = countFlag.ToString();
            text.GetComponent<Text>().text = mainTitle + mainText;
            PlayerPrefs.SetString($"{gameObject.name}", mainText);
        }
    }

    void OnTriggerEnter(Collider collider)
    {
        mainFlag = countFlag++;
    }
}
```

Клас ImageProgressBar

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.EventSystems;
using UnityEngine.UI;

public class ImageProgressBar : MonoBehaviour
{
    public GameObject interactiveObject;
            public UnityEvent onBarFilled;

    [Header("Custom Settiings")]
            public bool alwaysFacePlayer = true;
    public bool disableOnFill    = false;

            public float timeToFill = 1.0f;

            private Image progressBarImage = null;
            public Coroutine barFillCoroutine = null;

            void Start ()
            {
```

```
            progressBarImage = GetComponent<Image>();

            if(progressBarImage == null)
            {
                  Debug.LogError("There is no referenced image on this component!");
            }

            EventTrigger eventTrigger = interactiveObject.AddComponent<EventTrigger>();

            EventTrigger.Entry pointerEnter = new EventTrigger.Entry();
            pointerEnter.eventID = EventTriggerType.PointerEnter;
            pointerEnter.callback.AddListener((eventData) => { StartFillingProgressBar(); });
            eventTrigger.triggers.Add(pointerEnter);

            EventTrigger.Entry pointerExit = new EventTrigger.Entry();
            pointerExit.eventID = EventTriggerType.PointerExit;
            pointerExit.callback.AddListener((eventData) => { StopFillingProgressBar(); });
            eventTrigger.triggers.Add(pointerExit);
      }

      void Update()
      {
            if(alwaysFacePlayer)
            {
                  transform.LookAt(Player.GetPosition());
            }
      }

      void StartFillingProgressBar()
      {
            barFillCoroutine = StartCoroutine("Fill");
      }

      void StopFillingProgressBar()
      {
            StopCoroutine(barFillCoroutine);
            progressBarImage.fillAmount = 0.0f;
      }

      IEnumerator Fill()
      {
            float startTime = Time.time;
            float overTime = startTime + timeToFill;

            while(Time.time < overTime)
            {
                  progressBarImage.fillAmount = Mathf.Lerp(0, 1, (Time.time - startTime) /
timeToFill);
                  yield return null;
            }
```

```
                progressBarImage.fillAmount = 0.0f;

                if(onBarFilled != null)
                {
                        onBarFilled.Invoke();
                }

        if(disableOnFill)
        {
           transform.parent.gameObject.SetActive(false);
        }
            }
}
```

Клас EndGame

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class EndGame : MonoBehaviour
{
   private DateTime date1;
   public Text resultText;
   DateTime currResult;
   DateTime prevRes;

   public void Start()
   {
      Debug.Log("Datatime: " + prevRes);
      date1 = System.DateTime.Now;
      if (PlayerPrefs.GetString("parsedDate") == "")
      {
         prevRes = DateTime.Now;
      }
      else
      {
         Debug.Log(PlayerPrefs.GetString("parsedDate"));
         prevRes = DateTime.Parse(PlayerPrefs.GetString("parsedDate"));
         Debug.Log("Data: " + prevRes);
      }
   }

   public void Test1()
   {
      Debug.Log(date1);
      TimeSpan diff = DateTime.Now - date1;
```

```
        PlayerPrefs.SetString("Data", DateTime.Now.ToString());
        PlayerPrefs.SetString("MazeResult", diff.ToString());

        PlayerPrefs.SetString("parsedDate", diff.ToString());

        currResult = DateTime.Parse(diff.ToString());

        Debug.Log("First" + prevRes + " Second: " + currResult);

        if (prevRes < currResult)
        {
            Debug.Log("Bad Result");
            resultText.text = diff.ToString() + "\n" + "Bad Result";
        }
        else
        {
            Debug.Log("Good result!");
            resultText.text = diff.ToString() + "\n" + "Good Result";
        }

        Debug.Log("Datatime: " + currResult);
    }
    public void Test2()
    {
        Debug.Log("Hello from test2!");
    }
}
```

Клас MemoryScenesManager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MemoryScenesManager : MonoBehaviour
{
    public GameObject memoryScenes;
    public GameObject mapCanvas;
    public GameObject resultCanvas;
    public GameObject pointsResult;
    public void OpenMemoryScenes()
    {
        memoryScenes.SetActive(true);
    }

    public void ExitMemoryScenes()
    {
        memoryScenes.SetActive(false);
```

```
    }

    public void OpenMapCanvas()
    {
        mapCanvas.SetActive(true);
    }

    public void ExitMapCanvas()
    {
        mapCanvas.SetActive(false);
    }

    public void OpenResultCanvas()
    {
        resultCanvas.SetActive(true);
    }

    public void ExitResultCanvas()
    {
        resultCanvas.SetActive(false);
    }

    public void ShowPointsResult()
    {
        pointsResult.SetActive(true);
    }

    public void HidePointsResult()
    {
        pointsResult.SetActive(false);
    }
}
```

Клас ResultOfUser

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ResultOfUser : MonoBehaviour
{
    public Text resultText;
    void Start()
    {
        if (PlayerPrefs.GetString("Data") != "" && PlayerPrefs.GetString("MazeResult") != "")
        {
            PlayerPrefs.SetString("Res", PlayerPrefs.GetString("Res") + $"Date: {PlayerPrefs.GetString("Data")}
Result: {PlayerPrefs.GetString("MazeResult")} \n");
            resultText.text += PlayerPrefs.GetString("Res");
```

```
            PlayerPrefs.SetString("Data", "");
            PlayerPrefs.SetString("MazeResult", "");
        }
        else
        {
            resultText.text += PlayerPrefs.GetString("Res");
        }
    }
}
```

Клас RecordManager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace Recorder
{
    [RequireComponent(typeof(ScreenRecorder))]
    public class RecordManager : MonoBehaviour
    {
        ScreenRecorder recorder;
                    private void Start()
                    {
            recorder = GetComponent<ScreenRecorder>();
                    }

        public void StartRecord()
        {
            recorder.PrepareRecorder();
            StartCoroutine(DelayCallRecord());
        }
        private IEnumerator DelayCallRecord()
        {
            yield return new WaitForSeconds(0.1f);
            recorder.StartRecording();
        }


        public void StopRecord()
        {
            recorder.StopRecording();
            StartCoroutine(DelaySaveVideo());
        }
        private IEnumerator DelaySaveVideo()
        {
            yield return new WaitForSeconds(1);
            recorder.SaveVideoToGallery();
        }
    }
```

```
}
```

Клас GameController

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class GameController : MonoBehaviour
{
    private static GameController instance;

    public Sprite bgImage;
    public Text txt;

    public GameObject endGameCanvas;

    public GameObject[] objects;

    public Sprite[] puzzles;

    public List<Sprite> gamePuzzles;

    private List<Button> btns;

    public bool firstGuess, secondGuess;

    public int countGuesses;
    public int countCorrectGuesses;
    public int gameGuesses;

    public int firstGuessIndex, secondGuessIndex;

    private string firstGuessPuzzle, secondGuessPuzzle;

    void Awake()
    {
        instance = this;
    }

    void Start()
    {
        Debug.Log("Start game! " + EventSystem.current.name);
        instance.gamePuzzles = new List<Sprite>();
        instance.btns = new List<Button>();
        instance.firstGuess = false;
        instance.secondGuess = false;
```

```
        instance.GetButtons();
        instance.AddGamePuzzles();
        Shuffle(instance.gamePuzzles);
        instance.gameGuesses = instance.gamePuzzles.Count / 2;
    }

    void GetButtons()
    {
        for (int i = 0; i < objects.Length; i++)
        {
            btns.Add(objects[i].GetComponent<Button>());
            btns[i].image.sprite = bgImage;
        }
    }

    void AddGamePuzzles()
    {
        int looper = btns.Count;
        int index = 0;

        for (int i = 0; i < looper; i++)
        {
            if (index == looper / 2)
            {
                index = 0;
            }
            gamePuzzles.Add(puzzles[index]);
            index++;
        }
    }

    public static void PickAPuzzle(GameObject go)
    {

        if (!instance.firstGuess)
        {
            instance.firstGuess = true;
            Debug.Log("Error " + go.name);

            instance.firstGuessIndex = int.Parse(go.name);

            instance.firstGuessPuzzle = instance.gamePuzzles[instance.firstGuessIndex].name;

            instance.btns[instance.firstGuessIndex].image.sprite                                                    =
instance.gamePuzzles[instance.firstGuessIndex];
            instance.btns[instance.firstGuessIndex].GetComponent<Image>().raycastTarget = false;
        }
        else if (!instance.secondGuess)
        {
            instance.secondGuess = true;
```

```
            instance.secondGuessIndex = int.Parse(go.name);

            instance.secondGuessPuzzle = instance.gamePuzzles[instance.secondGuessIndex].name;

            instance.btns[instance.secondGuessIndex].image.sprite                                                    =
instance.gamePuzzles[instance.secondGuessIndex];
            instance.btns[instance.secondGuessIndex].GetComponent<Image>().raycastTarget = false;
            instance.countGuesses++;

            instance.StartCoroutine(CheckIfThePuzzleMatch());

            if (instance.firstGuessPuzzle == instance.secondGuessPuzzle)
            {
                Debug.Log("The puzzle match!");
            }
            else
            {
                Debug.Log("The puzzle don't match!");
            }
        }
    }

    static IEnumerator CheckIfThePuzzleMatch()
    {
        yield return new WaitForSeconds(1f);

        if (instance.firstGuessPuzzle == instance.secondGuessPuzzle)
        {
            yield return new WaitForSeconds(.5f);

            instance.btns[instance.firstGuessIndex].interactable = false;
            instance.btns[instance.secondGuessIndex].interactable = false;

            instance.btns[instance.firstGuessIndex].image.color = new Color(0, 0, 0, 0);
            instance.btns[instance.secondGuessIndex].image.color = new Color(0, 0, 0, 0);

            instance.btns[instance.firstGuessIndex].GetComponent<Image>().enabled = false;
            instance.btns[instance.secondGuessIndex].GetComponent<Image>().enabled = false;

            CheckIfTheGameIsFinished();
        }
        else
        {
            yield return new WaitForSeconds(.5f);
            instance.btns[instance.firstGuessIndex].GetComponent<Image>().raycastTarget = true;
            instance.btns[instance.secondGuessIndex].GetComponent<Image>().raycastTarget = true;
            instance.btns[instance.firstGuessIndex].image.sprite = instance.bgImage;
            instance.btns[instance.secondGuessIndex].image.sprite = instance.bgImage;
        }
```

```
        yield return new WaitForSeconds(.5f);

        instance.firstGuess = instance.secondGuess = false;
    }

    static void CheckIfTheGameIsFinished()
    {
        instance.countCorrectGuesses++;

        if (instance.countCorrectGuesses == instance.gameGuesses)
        {
            instance.endGameCanvas.SetActive(true);
            Debug.Log("Game finished!");
            Debug.Log("It took you " + instance.countGuesses + " many guess(es) to finish the game");
            instance.txt.text = "It took you " + instance.countGuesses + " many guess(es) to finish the game!";
            instance.countCorrectGuesses = 0;
            instance.countGuesses = 0;
        }
    }

    static void Shuffle(List<Sprite> list)
    {
        for (int i = 0; i < list.Count; i++)
        {
            Sprite temp = list[i];
            int randomIndex = Random.Range(i, list.Count);
            list[i] = list[randomIndex];
            list[randomIndex] = temp;
        }
    }
}
```